View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by UPCommons. Portal del coneixement obert de la UPC

TREBALL DE FI DE CARRERA

TÍTOL: Aplicació web per videoconferències

AUTOR: Janina Garrigós Sabaté

DIRECTOR: Toni Oller Arcas

DATA: 6 de març de 2007

Títol: Aplicació web per videoconferències

Autor: Janina Garrigós Sabaté

Director: Toni Oller Arcas

Data: 6 de març de 2007

Resum

Aquest treball descriu els passos seguits per desenvolupar una aplicació de videoconferències adequada per grans corporacions. Es descriu el disseny, l'arquitectura i la implementació utilitzats per aconseguir aquest objectiu.

El producte resultant s'anomena WiCat i es composa de varis elements que es comuniquen mitjançant una sèrie de protocols de xarxa. El WiCat Client és l'aplicació web amb la qual interactua l'usuari. El WiCat Server té com a funció principal la senyalització de les videoconferències entre els clients mitjançant el protocol SIP. Per comunicar client i servidor s'ha dissenyat un protocol propi basat en XML.

Els usuaris que utilitzin el WiCat Client disposaran d'una aplicació amb servei de presència, programació de videoconferències i capaç de realitzar trucades amb múltiples usuaris simultàniament.

WiCat es pot integrar amb les infraestructures d'emmagatzament d'informació existents en la organització, com poden ser directoris LDAP o bases de dades. Aquestes infraestructures s'utilitzen per obtenir la informació dels usuaris i utilitzar-la pel servei de presència i per les comunicacions SIP. Per fer consultes als LDAP o bases de dades, el WiCat Server utilitza una passarel·la basada en serveis Web.

Pel desenvolupament del WiCat Client i per la transmissió del contingut multimèdia s'han utilitzat les tecnologies oferides per Adobe Flash. El Flash Player per l'execució de l'aplicació web i el Flash Media Server per gestionar els fluxos multimèdia.

En aquest document s'expliquen les funcionalitats del plataforma de videoconferències WiCat, es descriu l'arquitectura de software utilitzada i finalment es dóna una visió sobre la implementació de cada mòdul.

Title: TFC/PFC Model

Author: Janina Garrigós Sabaté

Director: Toni Oller Arcas

Date: March, 6th 2007

Overview

This project describes the steps taken to develop a videoconference platform suitable for big corporations. The design, the architecture and the implementation used to reach this goal are described.

The resulting product is called WiCat and is composed by some elements that interconnect using a given protocol set. The WiCat Client is the web application a user uses to interact with. The WiCat server uses the SIP protocol to manage signalling between clients. Our own XML based protocol has been designed to allow communication between client and server.

The users of the WiCat client will get presence service and videoconference scheduling being able to do calls with multiple users simultaneously.

WiCat can be integrated with the corporation's existing information database infrastructures. For instance LDAP directories or traditional data bases. This infrastructure ara used to get client information used on presence services. To access this databases the WiCat server uses web services.

To develop the WiCat client and transmit the multimedia data streams a Macromedia Adobe solution is used. In concrete the Flash Player for the web application and the Flash Media Server to manage multimedia streams.

This document explains the functionalities of the videoconference platform WiCat, describes the software architecture used and finally a vision is given about the implementation used for every module.

ÍNDEX

ÍNDEX DE FIGURES I TAULES

INTRODUCCIÓ

Actualment moltes organitzacions opten per la utilització de sistemes de videoconferència sobre IP. L'increment del treball cooperatiu en aquest sector és un factor que afavoreix la utilització de la tecnologia WoiP[1]. La distància entre usuaris suposa la utilització d'un mitjà de comunicació entre ells.

La majoria d'organitzacions disposen d'infraestructures per indexar la informació sobre el seu personal, com poden ser bases de dades o directoris LDAP. La motivació d'aquest treball és aprofitar aquests sistemes d'emmagatzament per crear una aplicació orientada a la comunicació audiovisual entre el personal de la organització.

La mobilitat dels usuaris suposa un impediment a l'hora d'utilitzar softwares per videoconferència. Sovint s'utilitzen equips públics o no habituals, on possiblement no es disposi de privilegis d'administració per instal·lar software. Per suplir aquest inconvenient i aportar comoditat als usuaris, la millor opció és disposar d'un entorn web com a aplicació de videoconferència.

Actualment les aplicacions web per videoconferència són escasses i, la majoria, de pagament. L'objectiu d'aquest treball és oferir un sistema de videoconferències adequat per grans corporacions i amb una interfície web per la interacció amb l'usuari.

El sistema desenvolupat s'anomena WiCat i consta principalment d'una interfície web per l'usuari feta en Flash (WiCat Client) i d'un servidor desenvolupat en Java (WiCat Server). El client aporta comoditat als usuaris i el servidor realitza vàries funcions, entre elles afegeix una capa de senyalització per les trucades mitjançant el protocol SIP.

Les funcionalitats de les quals disposa WICAT són les següents: Proporciona un servei de presència d'usuaris. Pot mostrar la informació de contacte d'aquests usuaris. Ofereix la possibilitat de crear multi conferències. Permet programar videoconferències per una data determinada i amb els usuaris escollits.

Aquest treball es divideix en cinc parts, a continuació s'exposa el contingut de cadascuna:

En el primer capítol es descriuen les funcionalitats de l'aplicació amb l'ajuda d'un diagrama de casos d'ús.

El segon capítol mostra el disseny i l'arquitectura de software utilitzada per desenvolupar el sistema de videoconferències, a més s'afegeix un apartat on es pot veure la interacció entre els elements de l'arquitectura.

[1] Acrònim que s'utilitza per referir-se al vídeo i veu sobre IP.

En el tercer capítol s'explica la implementació de cada mòdul de l'arquitectura descrita i es mostra l'escenari de probes utilitzat.

La planificació del treball, les tasques realitzades i el temps empleat per a cadascuna es troben en el quart capítol.

Finalment, el cinquè capítol exposa les conclusions extretes de la realització del TFC. Es composa per tres apartats, els objectius assolits, l'impacte mediambiental i les conclusions personals.

CAPÍTOL 1. ESPECIFICACIÓ

El sistema de videoconferències WiCat segueix una especificació concreta. Al llarg d'aquest treball s'utilitza UML com a llenguatge de modelat orientat a objectes, s'incorporen diagrames de casos d'ús, d'interacció i de classes.

En aquest capítol s'enumeren les funcionalitats de WiCat, s'incorpora un diagrama de casos d'ús per veure les opcions que ofereix l'aplicació a l'usuari i finalment s'explica cada cas d'ús amb l'ajuda d'una taula CRC[1].

1.1. Funcionalitats del sistema de videoconferències

- **Videoconferència**. Realització de trucades de vídeo sobre IP recolzades per una capa de senyalització.

- **Programació de videoconferències**. L'usuari pot escollir la data i les característiques de la trucada, l'aplicació s'encarrega d'avisar-lo quan és l'hora.

- **Llista de contactes**. El WiCat Client ofereix una llista d'usuaris amb els corresponents indicadors de presència (usuari connectat o no connectat) que s'actualitza quan hi ha algun canvi.

- **Realització de multi conferències**. És possible iniciar una videoconferència amb més d'un usuari simultàniament.

- **Entorn web**. L'aplicació de l'usuari és de tecnologia web, permetent la seva mobilitat.

- **Portabilitat**. L'aplicació de l'usuari es pot executar en qualsevol equip que disposi d'un navegador web i el *plugin* de Flash. El WiCat Server només precisa d'una màquina virtual de Java pel seu funcionament.

1.2. Casos d'ús de l'usuari

En el diagrama UML següent (**¡Error! No se encuentra el origen de la referencia.**) es mostren les opcions que tenen els usuaris de l'aplicació.

[1] Class Responsibility Collaborator (Consultar els ACRÒNIMS)

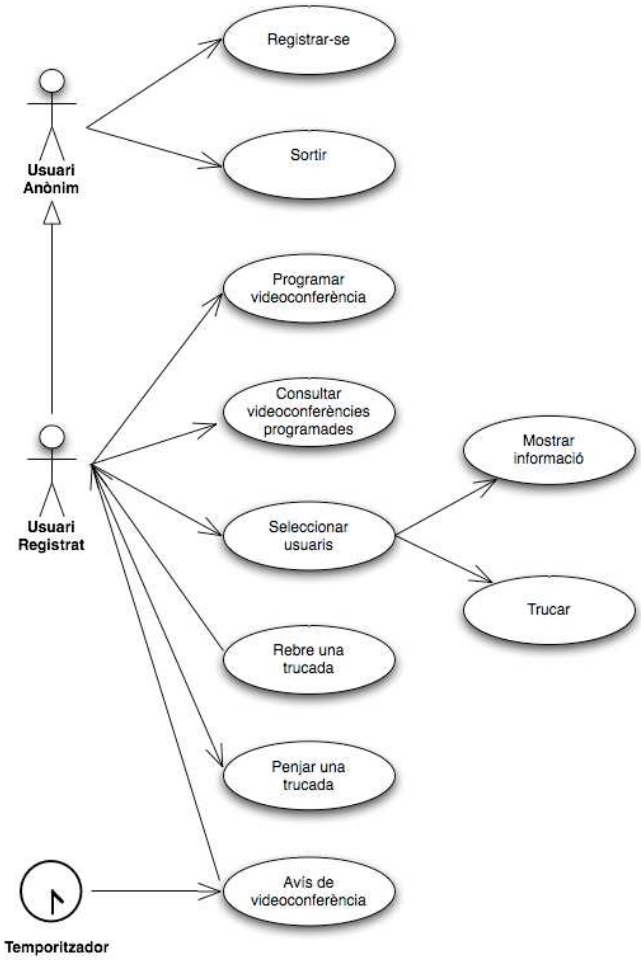


Fig. 1.1 Casos d'ús de l'usuari

Les taules següents descriuen cada cas d'ús, el seu flux normal i les condicions que s'han de donar perquè es compleixin.

Taula 1.1 Descripció dels casos d'ús de l'usuari

Nom	Registrar-se
Descripció	Registrar l'usuari en l'aplicació.
Actors	Usuari Anònim.
Flux normal	1. L'usuari introdueix el nom d'usuari i contrasenya. 2. Clica el botó "Enviar" .
Entrada	Credencials de l'usuari.
Sortida	Autenticació contra el directori LDAP, canvi a estat *online* si les credencials són correctes.

Nom	Sortir
Descripció	Desconnectar l'usuari de l'aplicació.
Actors	Usuari Anònim o usuari Registrat.
Flux normal	1. L'usuari clica el botó "sortir".
Flux alternatiu	L'usuari tanca el navegador web o clica el botó de refresc de la pàgina.
Sortida	Desconnexió de l'usuari, canvi a estat *offline*.

Nom	Programar videoconferència
Descripció	Programar una videoconferència per una data determinada.
Actors	Usuari Registrat.
Pre condicions	L'usuari està registrat.
Flux normal	1. Triar la data i la hora. 2. Escollir els usuaris que hi participaran. 3. Introduir una descripció.
Entrada	Informació de la videoconferència.
Sortida	Programació de la videoconferència per part del gestor de videoconferències.

Nom	Consultar videoconferències.
Descripció	Consultar la informació de les videoconferències programades en les que participarà l'usuari.
Actors	Usuari Registrat.
Pre condicions	L'usuari està registrat i pertany a alguna videoconferència programada.
Flux normal	1. Triar el panell "consultar videoconferències". 2. Clicar el botó "consultar". 3. És possible triar una de les dates per veure la informació de la videoconferència.
Flux alternatiu	Si hi ha una videoconferència activa on l'usuari està convidat: 1. Es mostra una finestra d'avís. 2. L'usuari pot clicar el botó d'acceptar per unir-se a la videoconferència o el de declinar per veure la informació de les videoconferències.
Sortida	Llista de les dates en les que hi ha videoconferències.

	Panell on es mostra tota la informació seleccionant una data.

Nom	Mostrar informació.
Descripció	Mostra la informació de contacte dels usuaris seleccionats.
Actors	Usuari Registrat.
Pre condicions	L'usuari està registrat.
Flux normal	1. Seleccionar un o més usuaris. 2. Clicar el botó d'informació.
Entrada	Usuaris seleccionats.
Sortida	Informació de contacte de cada usuari.

Nom	Trucar
Descripció	Convida a un o més usuaris a iniciar una videoconferència.
Actors	Usuari Registrat.
Pre condicions	L'usuari està registrat. Els usuaris trucats estan en estat *online*.
Flux normal	1. Seleccionar un o més usuaris connectats. 2. Clicar el botó de trucar. 3. Introduir una descripció.
Entrada	Usuaris seleccionats.
Sortida	Si accepta la invitació un usuari com a mínim, s'inicia la videoconferència automàticament.

Nom	Rebre trucada
Descripció	Quan es rep una trucada es mostra una finestra d'alerta amb la informació de la trucada.
Actors	Usuari Registrat.
Pre condicions	L'usuari està registrat.
Flux normal	1. Es mostra la informació de la videoconferència. 2. L'usuari pot clicar el botó d'acceptar o de declinar.
Post condicions	Mentre l'usuari estigui realitzant una videoconferència, no en pot iniciar cap altra.

Entrada	Invitació a videoconferència.
Sortida	Si l'usuari accepta s'inicia la videoconferència.

Nom	Penjar una trucada
Descripció	Quan l'usuari està realitzant una videoconferència pot abandonar-la clicant el botó de penjar.
Actors	Usuari Registrat.
Pre condicions	L'usuari està realitzant una videoconferència.
Flux normal	1. L'usuari clica el botó de penjar. 2. L'usuari torna a la pàgina principal.
Flux alternatiu	La resta d'usuaris de la videoconferència cliquen el botó de penjar.
Sortida	Es para l'enviament d'àudio i vídeo.

Nom	Avís de videoconferència
Descripció	El gestor de videoconferències avisa a l'usuari quan és l'hora d'iniciar una videoconferència programada.
Actors	Usuari Temporitzador.
Pre condicions	Existeix una videoconferència programada per l'usuari, en la data actual.
Flux normal	1. Es mostra una finestra d'avís. 2. L'usuari pot clicar el botó d'acceptar o de declinar.
Entrada	Videoconferència programada.
Sortida	Si l'usuari accepta iniciar la videoconferència, el gestor de videoconferències s'encarrega d'avisar a la resta d'usuaris implicats, com en una trucada normal.

CAPÍTOL 2. ARQUITECTURA

En aquest capítol es descriu l'arquitectura utilitzada per construir la plataforma per videoconferències WiCat. Ha de ser adequada pel personal d'una empresa i recolzada per un protocol de senyalització.

En el primer apartat s'exposen les opcions tecnològiques per poder implementar les funcionalitats desitjades. Després d'escollir la alternativa més adequada es mostra el disseny final de l'arquitectura.

En els següents apartats s'explica la funció de cada mòdul, classificats segons l'àmbit al que pertanyen. Al final del capítol hi ha una secció on es pot veure la interacció entre la interfície d'usuari i els mòduls encarregats de la gestió i senyalització de videoconferències.

2.1. Disseny

Per desenvolupar les funcionalitats citades en el capítol anterior s'han estudiat varies alternatives. Cadascuna suposa l'ús de diverses tecnologies i la seva posterior integració. Degut a aquesta dificultat s'ha hagut de realitzar un estudi de viabilitat per cada opció.

2.1.1. Opcions tecnològiques

En una arquitectura de veu o vídeo sobre IP, es requereixen protocols que controlin la senyalització. Aquests protocols s'usen per establir connexions entre terminals, o entre un terminal i un element intermig. Regulen, entre d'altres coses, la presència, és a dir, l'entrada i sortida de nodes de la xarxa. Existeixen dos protocols de senyalització adequats per l'aplicació de videoconferències:

H323: Aquest és un protocol de la ITU-T[1] que s'utilitza per proveir de comunicació audiovisual a qualsevol xarxa de paquets. Està format per un conjunt d'estàndards que no només defineixen el model bàsic de trucada, sinó que ofereixen molts serveis complementaris relacionats amb la comunicació multimèdia.

SIP: Session **I**nitiation **P**rotocol, és un protocol de la capa d'aplicació usat per crear, modificar i finalitzar sessions entre un o varis participants. L'última versió de l'especificació és l'RFC 3261 [3] de la IETF[2]. S'utilitza principalment com a

[1] ITU Telecommunication Standardization Sector (Consultar els ACRÒNIMS)
[2] Internet Engineering Task Force (Consultar els ACRÒNIMS)

protocol de senyalització per veu sobre IP, encara que també es fa servir en altres àmbits, com en jocs multi jugador.

El protocol escollit per la senyalització de les videoconferències ha estat el SIP. El principal avantatge que proporciona és la seva simplicitat i el fet que està basat en text, igual que l'HTTP. Un inconvenient de l'H323 és que defineix un esquema centralitzat, basat en la xarxa telefònica i no permet la mobilitat dels usuaris.

La interfície d'usuari, era important que fos web. Un dels avantatges que ofereix aquesta tecnologia és que no requereix instal·lacions per part de l'usuari. A més és independent de la plataforma i permet la mobilitat dels usuaris, ja que només es requereix un navegador web per utilitzar-la.

Com a llenguatge de programació per implementar l'aplicació, es va escollir el Java. Aquest llenguatge proporciona orientació a objectes, aspecte important a l'hora de desenvolupar aplicacions complexes ràpidament. A més és portable i disposa d'una llibreria completa per implementar les funcions del protocol SIP, la API JAIN SIP [6] .

Per poder realitzar una videoconferència en un entorn web i recolzar-la amb el protocol SIP es van proposar tres solucions:

- Desenvolupar un Applet Java que implementi totes les funcionalitats de l'aplicació. La interacció amb l'usuari, el transport del contingut multimèdia i la senyalització SIP.

- Desenvolupar un Applet Java que implementi totes les funcionalitats, excepte la visualització dels fluxos multimèdia, per aquest fi s'utilitzaria un reproductor extern.

- Desenvolupar un client web amb Flash que implementi la reproducció i transmissió dels fluxos multimèdia, i delegar la senyalització SIP a un element extern desenvolupat en Java.

La primera i segona opció tenen un inconvenient important, la utilització d'un Applet comporta un pes excessiu de l'aplicació del client. A més de la dificultat per establir connexions des d'aquest Applet, ja que té moltes restriccions de seguretat[1].

Un altre inconvenient de la primera alternativa és que imposa l'ús del *framework* JMF[2][9] per la transmissió i reproducció dels fluxos multimèdia. Aquesta llibreria és mol complexa i ineficient. A més la última actualització de la API data del novembre de 2004, dada que demostra un abandonament per part dels desenvolupadors de Java Sun.

[1] Applet Sandbox. Més informació a la URL: http://www.securingjava.com/chapter-two/
[2] Java Media Framework (Consultar els ACRÒNIMS)

La tercera opció té l'avantatge de gestionar la transmissió i reproducció dels fluxos multimèdia d'una manera simple i fàcil de programar. Un dels inconvenients és la impossibilitat d'integrar una pila SIP en el client. A més és necessari l'ús d'un element extern encarregat de commutar i retransmetre els fluxos multimèdia dels usuaris.

Després d'analitzar els avantatges i inconvenients de cada opció es va optar per la utilització de Flash. Aquesta elecció comporta l'ús d'un servidor Flash que s'encarregui de gestionar els fluxos multimèdia dels usuaris. Per aquest fi existeixen dues opcions.

Red5 [22] : Servidor Flash de codi obert, escrit en Java. Suporta *streaming* d'àudio i vídeo, objectes compartits i publicació de fluxos en viu.

Flash Media Server[1]: Servidor Flash propietari d'Adobe. Té les mateixes funcionalitats que el Red5 i n'afegeix moltes altres com, per exemple, connexió a serveis externs com serveis web, XMLSockets,etc.

Per la realització del treball es va escollir el Flash Media Server. La raó principal és que el servidor Red5 està en fase de desenvolupament, la versió disponible és la 0.6, no és prou fiable ni complet per utilitzar-lo en aquest treball. S'ha utilitzat la versió per desenvolupadors del Flash Media Server, una versió gratuïta però limitada a deu connexions. Pel desenvolupament del treball deu connexions (deu clients realitzant videoconferència) són suficients.

La utilització de la plataforma de Flash Media Server té un inconvenient, no és possible realitzar una comunicació multimèdia entre els clients Flash i altres clients SIP. Això és degut a que Flash utilitza per la transmissió de fluxos d'àudio i vídeo el protocol propietari RTMP[2], en canvi els clients SIP utilitzen RTP[3]. Per tant, seria necessari afegir un element al sistema que fos capaç de traduir paquets RTP a RTMP i a l'inrevés. Això queda fora de la intenció d'aquest TFC per la seva complexitat, però no es descarta com a treball futur.

2.1.2. Disseny de l'arquitectura

El disseny de l'arquitectura general s'ha fet tenint en compte la simplicitat d'implementació, les necessitats dels clients als quals va dirigida l'aplicació i les limitacions que suposa l'elecció d'un client web.

Segons la funcionalitat de cada mòdul s'han dividit en tres grups, en la Fig. 2.1 es poden veure diferenciats per color.

[1] Consultar ANNEX B
[2] Real Time Messaging Protocol (Consultar els ACRÒNIMS)
[3] Real-time Transport Protocol (Consultar els ACRÒNIMS)

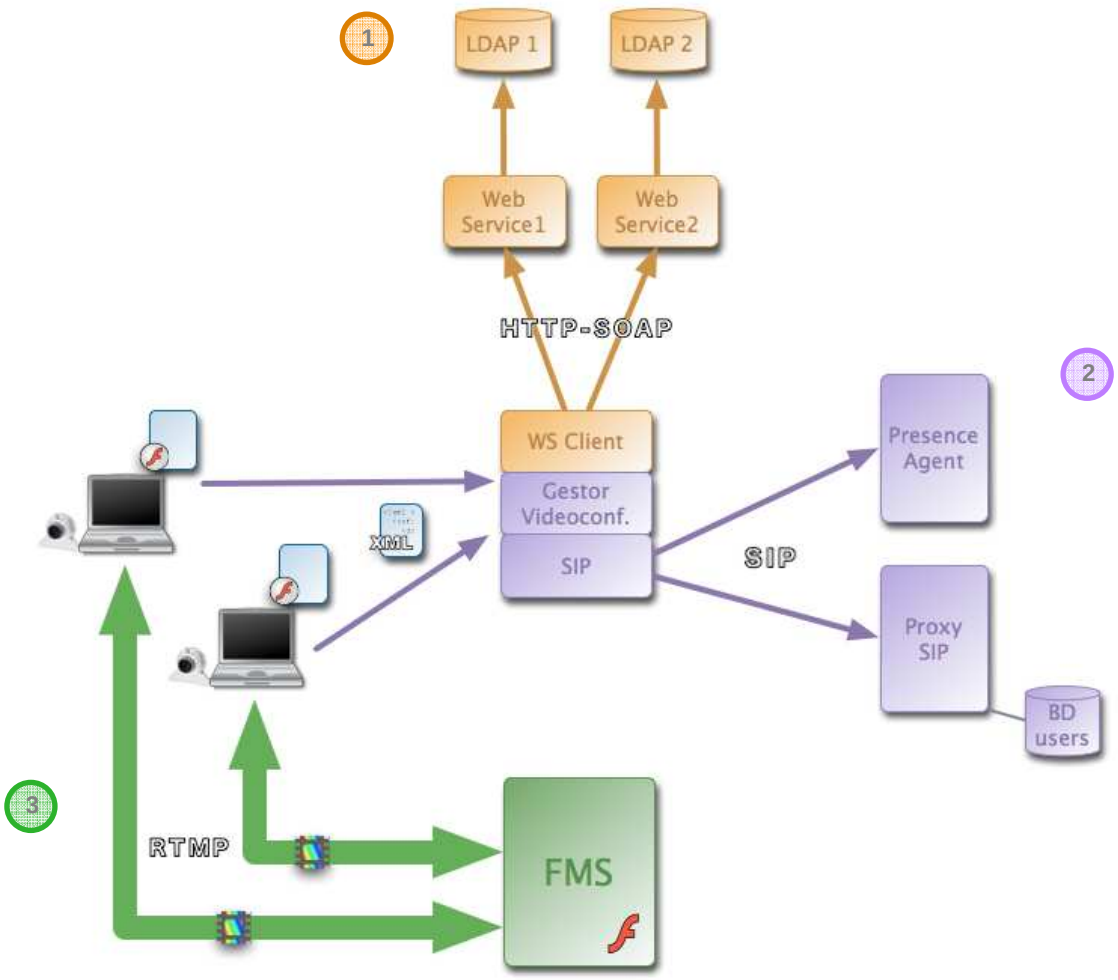


Fig. 2.1 Arquitectura general del sistema

Gestió d'usuaris i equips de videoconferència:

El sector numero 1 mostra els mòduls del sistema utilitzats per obtenir la informació dels usuaris i dels seus equips de videoconferència. Aquests mòduls són els següents:

LDAP[1]: Directori que emmagatzema la informació sobre els usuaris i els equips de videoconferència.

Servei Web: Element accessible des d'Internet que fa peticions al directori LDAP.

Client del Servei Web: Aquest element està integrat en el WiCat Server. La seva funció és fer peticions al Servei Web mitjançant HTTP-SOAP[2].

[1] Lightweight Directory Access Protocol (Consultar els ACRÒNIMS)
[2] HyperText Transfer Protocol – Simple Object Access Protocol (Consultar els ACRÒNIMS)

Senyalització i gestió de videoconferències:

La part numero 2 mostra els diferents elements i protocols implicats en l'establiment de videoconferències i multi conferències. També la gestió d'aquestes videoconferències (programació, avís,etc). Consta dels següents elements:

> **Client Flash** : Element web amb el qual interacciona l'usuari i que es comunica amb la resta mitjançant un protocol propi basat en XML.
>
> **Gestor de videoconferències**: Element que s'encarrega d'emmagatzemar les videoconferències programades pels usuaris i de gestionar la seva realització.
>
> **Pila SIP** : Aquest mòdul s'encarrega de senyalització de les trucades. Rep peticions per part de l'usuari i les tradueix a peticions SIP.
>
> **Presence Agent**: És l'encarregat de gestionar la presència dels usuaris.
>
> **Proxy SIP:** Fa d'element intermig entre la resta d'elements SIP. Registra aquests elements en la seva base de dades. És l'encarregat de rebre peticions SIP i reenviar-les cap al seu destinatari

Transmissió de contingut multimèdia:

Els elements i protocol que realitzen aquesta funció corresponen al sector número 3. Són els utilitzats per transmetre l'àudio i vídeo entre els diferents usuaris d'una videoconferència.

> **Client Flash**: Client web basat en Flash que permet visualitzar els fluxos d'àudio/vídeo de les videoconferències.
>
> **Flash Media Server**: Software dedicat a commutar i redireccionar els fluxos multimèdia dels usuaris mitjançant el protocol propietari RTMP.

2.2. Gestió d'usuaris i equips de videoconferència

Tota corporació que utilitzi la VoIP o el vídeo sobre IP per la comunicació dels seus empleats, necessita alguna forma d'emmagatzemar la informació dels terminals VoIP i de relacionar-los amb les persones que els utilitzen. En aquest treball s'ha optat per l'ús de directoris LDAP, però és possible integrar l'aplicació amb les infraestructures existents en la organització, com poden ser bases de dades. Això ho permet la utilització de serveis Web, utilitzats per fer consultes a aquests sistemes d'emmagatzament.

2.2.1. LDAP

El *Lightweight Directory Access Protocol* (LDAP) és un protocol que permet l'accés a un servei de directori ordenat i distribuït que emmagatzema informació en un entorn de xarxa. Normalment s'utilitza per guardar informació de contacte i sobretot informació de *login* (usuari i contrasenya).

La utilització de serveis Web i de directoris LDAP permet a les organitzacions triar entre varies combinacions per indexar la informació sobre els usuaris i els equips de videoconferència. Seguidament se'n mostren tres exemples.

- Utilitzar el mateix LDAP per guardar la informació dels usuaris i dels equips de videoconferència.
- Fer servir un LDAP per emmagatzemar la informació dels usuaris i un altre pels equips.
- Tenir un LDAP a la seu central amb la informació dels equips de videoconferència i, a cada sucursal, utilitzar un LDAP pels usuaris.

La primera i segona opció es veuen representades en la Fig. 2.2.

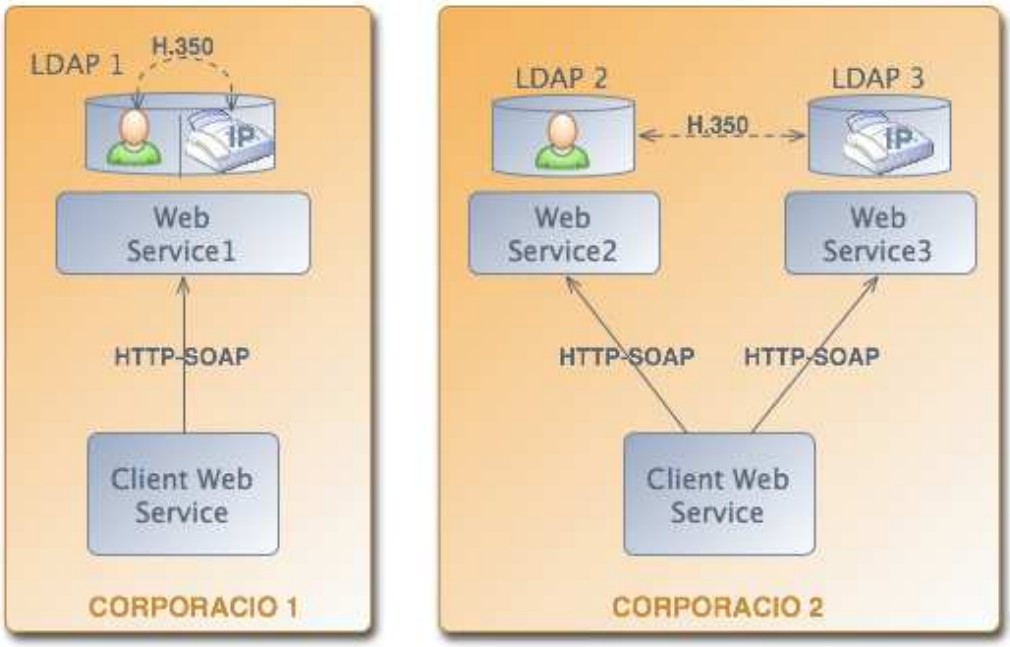


Fig. 2.2 Relació entre usuaris i equips de videoconferència

Totes aquestes opcions són vàlides a l'hora de gestionar la informació d'usuaris i d'equips de videoconferència. Això és possible gràcies a l'estàndard de la ITU-T H.350 (Directory Services Arquitecture for Multimedia Conferencing)[27] .

L' H.350 descriu l'arquitectura de serveis de directori per conferències multimèdia utilitzant LDAP. Aquest estàndard defineix la manera de relacionar cada persona d'un directori amb el seu corresponent equip de videoconferència, emplaçat en el mateix LDAP o en un altre.

2.2.2. Serveis Web

Les consultes a cada LDAP de la corporació es realitzen mitjançant serveis web. Per cada LDAP al qual es vulguin fer consultes, s'instal·larà un servei web amb la informació necessària per accedir a aquest LDAP.

Els Web services o serveis web són softwares destinats a la comunicació entre màquines a través d'una xarxa. Normalment es tracta d'una API que ofereix certs serveis. Es pot entendre com una caixa negra que conté un cert algoritme, que compleix una funció clara i a la qual s'hi pot accedir des d'Internet.

Alguns dels avantatges que aporten els serveis web són els següents: L'ús d'HTTP per la comunicació permet travessar tallafocs. Permeten la interoperabilitat entre aplicacions independentment de la plataforma on s'executin i la forma com s'hagin creat aquestes aplicacions. Alliberen les aplicacions de carga de CPU. Hi ha la possibilitat de combinar-se entre ells per realitzar una tasca determinada.

Els serveis web també comporten alguns inconvenients. No tenen memòria, reben una petició, la processen i tornen la resposta. El mecanisme de serialització és pesat, ja que es basa en XML. Tenen problemes de seguretat, sobretot per l'autenticació, qualsevol persona que accedeixi al arxiu WSDL[1] on es defineixen els mètodes que ofereix, pot fer us del servei web sense permís.

Els serveis web permeten utilitzar una interfície comuna per fer consultes en totes les organitzacions independentment de la complexitat dels sistemes d'informació dels que disposin. Per exemple, enlloc d'un directori LDAP les corporacions podrien utilitzar el servei web per accedir a una base de dades i l'aplicació que consulta el servei web no s'hauria de modificar en absolut.

Els serveis web implementats per l'aplicació aporten una sèrie d'avantatges administratius per les organitzacions:

- La organització no ha de facilitar cap contrasenya d'accés als usuaris que vulguin fer consultes al LDAP. Aquesta contrasenya pot ser incorporada fàcilment en el servei web pel personal autoritzat.
- Es pot obtenir fàcilment la informació de tots els LDAP's de la organització consultant cada servei web.
- Cada sucursal de l'empresa pot restringir l'accés a certes entrades "privades" de l'LDAP mitjançant el servei web.

Les operacions bàsiques de què disposa el servei web són les següents:

- Llistar tots els atributs del directori. Per exemple mail, telèfon,etc.
- Buscar una entrada per l'atribut i valor. Per exemple buscar les entrades que tinguin per *mail* persona@domini.com. També és possible fer una cerca de totes les entrades que continguin l'atribut *mail*.

[1] Web Services Definition Language (Veure els ACRÒNIMS)

- Fer una cerca de totes les entrades de persones que tinguin equip de videoconferència. Aquesta operació retorna l'entrada LDAP de la persona i els atributs corresponents a l'equip de videoconferència.

2.2.3. Client del Servei Web

L'accés a les operacions dels serveis web es realitza mitjançant un client de servei web. Aquest client està incorporat en el WiCat Server, d'aquesta manera els usuaris de l'aplicació web s'hi poden comunicar a través del protocol propi basat en XML. El client del servei web es pot configurar perquè accedeixi a un o més serveis web, segons si es vol obtenir informació d'un o més LDAP's.

Aquest client fa crides SOAP al servei web sobre HTTP. Això és un avantatge a l'hora de consultar serveis web que resideixin en organitzacions diferents que acostumen a disposar d'un tallafocs. Aquestes crides passaran pel tallafocs sense ser interceptades ja que es fan al port 80.

2.3. Senyalització i gestió de videoconferències

En els últims anys les aplicacions de veu sobre IP i més recentment les de vídeo sobre IP, s'han desplegat notablement. L'increment de l'ample de banda i el desenvolupament de nous *codecs* han millorat molt la qualitat subjectiva tant de les trucades de veu sobre IP com les de vídeo sobre IP.

Existeixen moltes d'aplicacions de WoIP que usen SIP (RFC 3261) per la senyalització, tot seguit s'enumeren alguns exemples de user agents SIP que permeten el vídeo sobre IP:

- **Linphone**[1]: Softphone de codi obert, permet comunicacions d'àudio, vídeo i missatgeria instantània.

- **eyeBeam**[2]: Softphone per realitzar videoconferències senyalitzades amb SIP. Disponible per Windows i Mac Os. La versió completa inclou àudio, vídeo, missatgeria instantània i presència. Té un cost de 60 $.

- **MXM 3.0**[3]: Conjunt de dues aplicacions. L'vPoint, client de videoconferència. L'IPNexus, eina de missatgeria instantània i de compartició de dades. Suporta SIP i H323. És orientat a l'àmbit empresarial i de pagament.

El disseny de l'aplicació, està fet pensant en la compatibilitat amb les aplicacions que implementen SIP com a protocol de senyalització per vídeo

[1] Linphone URL: http://www.linphone.org/
[2] eyeBeam URL: http://www.inphonex.com/support/eyebeam.php?PHPSESSID=f6987939e2dd694eac738e255933e3ee
[3] MXM 3.0 URL: http://www.34t.com/box-docs.asp?doc=680

sobre IP, com les mencionades. Per poder-ho aconseguir s'hauria d'afegir un altre element al sistema, com s'ha explicat en l'apartat 2.1.1. Però a falta d'aquest, tots els elements implicats en la senyalització han estat dissenyats seguint l'estàndard SIP.

El següent esquema (Fig. 2.3) mostra els mòduls del sistema implicats en la senyalització i la gestió de les videoconferències.

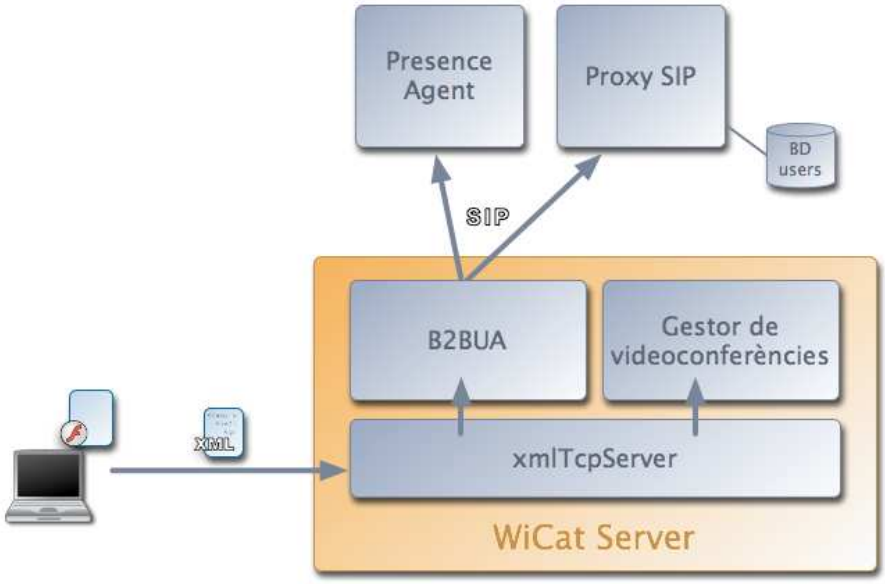


Fig. 2.3 Mòduls encarregats de la senyalització i gestió de videoconferències

El client Flash es comunica amb el WiCat Server mitjançant un protocol propi basat en XML. Aquest protocol permet transmetre missatges tant de senyalització, com de control. Els missatges de senyalització corresponen a peticions i respostes SIP i els de control són la resta de missatges necessaris per la gestió de videoconferències. L'xmlTcpServer processa aquests missatges i crida les operacions necessàries del B2BUA[1] de l'usuari o del Gestor de Videoconferències, segons correspongui.

En els següents apartats es dóna una visió més detallada de les operacions que permet cada mòdul implicat en la senyalització i gestió de videoconferències.

2.3.1. Client Web

El client web (WiCat Client) està desenvolupat en Flash. Es comunica amb el servidor mitjançant el protocol propi mencionat i realitza les operacions necessàries segons els missatges rebuts. Al final d'aquest capítol (2.5 Diagrames d'interacció entre els diferents mòduls) s'explica la interacció del WiCat Client amb la resta d'elements amb l'ajuda de diagrames.

[1] Back-to-Back User Agent (Veure els ACRÒNIMS)

2.3.2. B2BUA

El Back-to-Back User Agent (B2BUA) fa d'intermediari entre el client web i el món SIP. La seva funció és traduir les peticions del WiCat Client, fetes mitjançant un protocol basat en XML, a missatges SIP. Aquest element processa peticions SIP *request* com un User Agent Server (UAS), i també actua com un User Agent Client (UAC), determina com s'han de respondre les peticions rebudes i com iniciar una trucada.

La diferència principal entre un B2BUA i un UA[1] és que l'usuari interactua directament amb l'UA, que tradueix les seves ordres a peticions SIP *request*. En canvi el B2BUA es troba en un nivell inferior, entre l'aplicació de l'usuari i el proxy SIP. El motiu d'utilitzar aquest intermediari és que el client Flash amb el qual l'usuari interacciona no disposa d'una pila SIP per poder funcionar com un UA. Necessita un element intermig que realitzi les funcions de senyalització que ell no suporta.

En el següent diagrama es pot observar la comunicació de dos UA SIP:

Fig. 2.4 INVITE entre dos User Agents SIP

En el següent diagrama es pot observar la comunicació de dos clients Flash amb els seus corresponents B2BUA:

Fig. 2.5 INVITE entre dos usuaris de l'aplicació

2.3.3. Proxy SIP / REGISTRAR

Un SIP proxy és un element que dirigeix peticions SIP *request* cap als UAS i respostes SIP *response* cap als UAC. Una *request* pot travessar varis proxys abans d'arribar al UAS final.

[1] User Agent SIP (Veure ACRÒNIMS)

Un proxy SIP pren decisions d'enrutament, modificant la petició SIP *request* abans de reenviar-la cap al següent element. Les *responses* passaran a través dels mateixos proxys els quals ha travessat la *request*, però en l'ordre invers.

Una altra funció dels proxys SIP és el registre. El registre serveix perquè el proxy sàpiga la localització d'un usuari per tal de reenviar-li les *request* que rebi per ell. Els missatges REGISTER associen la SIP URI de l'user agent que l'envia amb l'equip on l'usuari es troba. El proxy emmagatzema aquesta associació en una base de dades.

Alguns proxys SIP poden requerir autenticació per part del user agent. Si és així, abans de registrar l'usuari en la seva base de dades li enviarà un desafiament perquè aporti credencials.

2.3.4. Agent de presència (Presence Agent)

El Presence Agent és un dels elements més importants del sistema. S'encarrega de gestionar la presència d'usuaris (l'entrada i sortida de l'aplicació). Aquesta funcionalitat permet saber en tot moment quins usuaris poden rebre trucades i quins no.

A més de la funcionalitat de presència aquest mòdul ha servit per la realització de videoconferències de més de dos participants. Utilitzant aquest element es simplifica la senyalització de les multi conferències, en el capítol següent s'explica i es justifica el mecanisme utilitzat.

Les dues funcions del Presence Agent es realitzen de manera molt similar. El Presence Agent rep subscripcions dels B2BUA's i els envia notificacions quan es produeix algun event. Existeixen dos tipus de subscripcions, les subscripcions a presència i les subscripcions a trucades.

El següent esquema (Fig. 2.6) [1] mostra un exemple de subscripció d'usuaris al Presence Agent. L'esquema serveix tant pel cas de les subscripcions a presència com a trucades.

[1] En la Fig. 2.6 s'han omes les respostes SIP 200 OK per evitar la illegibilitat

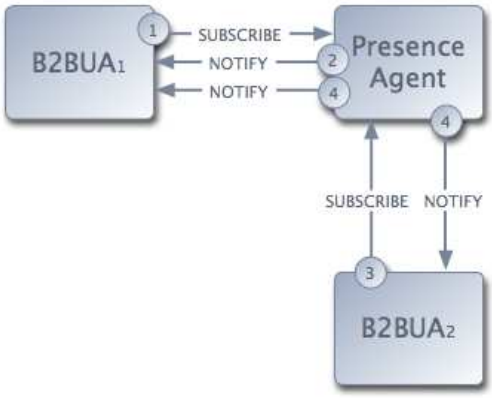


Fig. 2.6 Subscripció al Presence Agent

El B2BUA1 es subscriu primer i rep una notificació, després es subscriu el B2BUA2 i el Presence Agent notifica a tots dos usuaris.

2.3.5. Gestor de videoconferències

Aquest mòdul forma part del WiCat Server. És l'encarregat de gestionar les videoconferències programades pels clients. Realitza vàries funcions:

- Emmagatzema les videoconferències reservades pels clients.

- Quan un client ho consulta, li envia la informació sobre les videoconferències programades en les que està convidat.

- Avisa al client corresponent quan és l'hora d'iniciar una videoconferència programada per ell. A aquest client se l'anomena "màster de la videoconferència".

- Rep l'ordre del màster de la videoconferència per iniciar-la i avisa a la resta d'usuaris implicats perquè s'hi uneixin.

- Emmagatzema temporalment les videoconferències que hi ha en curs.

- Elimina una videoconferència quan el màster d'aquesta en surt.

- Quan un client consulta les videoconferències programades, comprova si n'hi ha alguna en curs i si l'usuari en qüestió hi està convidat. Si és així el convida a unir-se a la videoconferència.

2.4. Gestió de contingut multimèdia

L'aplicació web amb la qual interactua l'usuari està desenvolupada en Adobe Flash. L'entorn d'edició multimèdia de Adobe Flash permet crear aplicacions interactives i animacions fàcilment. A més incorpora una API anomenada ActionScript que, des de la versió 2.0 passa de ser programació estructural, a programació orientada a objectes.

El Flash és adequat per l'aplicació web sobretot perquè permet el treball amb vídeo. És possible integrar un arxiu de vídeo a la pel·lícula de Flash, carregar progressivament un vídeo des d'una font externa i també realitzar streaming. La tercera opció és la utilitzada en aquest Treball ja que permet capturar vídeo d'una webcam, l'àudio d'un micròfon, enviar el flux a través d'Internet i rebre els fluxos d'altres usuaris.

Apart de les facilitats de programació i les tècniques de vídeo que aporta Flash, té un avantatge molt important respecte d'altres tecnologies. El 98% dels navegadors web tenen instal·lat el Adobe Flash Player[1], per tant l'usuari no necessita res més per començar a utilitzar l'aplicació de videoconferències.

2.4.1. Flash Media Server

Per poder aprofitar la capacitat d'*streaming* de Flash és necessària la incorporació del Flash Media Server al sistema de videoconferències.

La plataforma del Flash Media Server està formada per dues parts: el servidor i el Flash Player. Les aplicacions creades amb aquesta plataforma estan composades per un client de Flash (fitxer SWF) que s'executa en el Flash Player, i un component servidor que es comunica amb el client.

El servidor i el client Flash es comuniquen a través d'una connexió persistent fent servir RTMP.

En un escenari típic, un servidor Web entrega l'arxiu SWF al navegador mitjançant HTTP. Un cop reproduint-se, la pel·lícula de Flash fa servir RTMP per establir la connexió amb el Flash Media Server, permetent un flux de dades continu entre client i servidor (**¡Error! No se encuentra el**

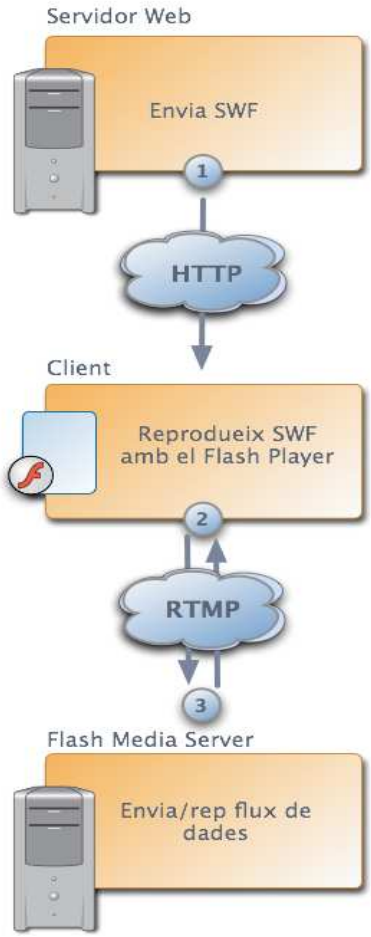


Fig. 2.7 Arquitectura del Flash Media Server

[1] Dada extreta de l'estudi *Flash Player Penetration.* URL: http://www.adobe.com/products/player_census/flashplayer/

origen de la referencia.).
El flux mencionat es composa d'àudio, vídeo i/o missatges de dades sincronitzats, que es transmeten des del client cap al servidor o del servidor cap al client. Aquests fluxos fan servir un model de publicació i subscripció que simplifica la programació. Un flux publicat pot ser reproduït a temps real (adequat per aplicacions de xat, com en el cas d'aquest treball), o gravat i reproduït en un altre moment. El diagrama següent mostra un exemple de flux reproduït a temps real.

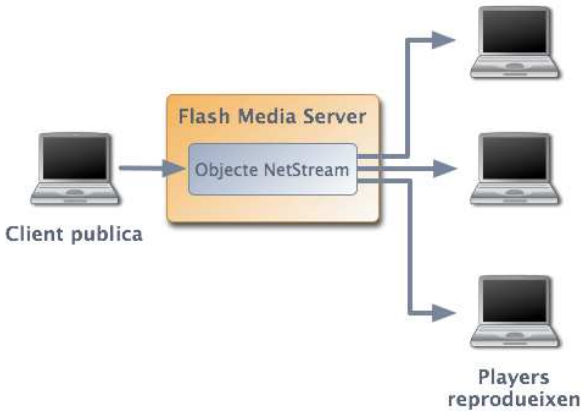


Fig. 2.7 Reproducció d'un flux d'àudio/vídeo a temps real

L'objecte NetStream representa el flux d'àudio i vídeo del client. Aquest flux s'allotja en el Flash Media Server perquè altres clients SWF el puguin reproduir en viu.

2.4.2. Flash Vídeo

El format multimèdia utilitzat per la transmissió sobre RTMP és el FLV[1]. Aquest format és propietari d'Adobe Flash i té com a principal avantatge que no és necessari cap descodificador addicional per visualitzar-lo. Gràcies a això és possible reproduir vídeos FLV en qualsevol navegador de qualsevol sistema operatiu que disposi del *plugin* de Flash (Flash Player).

Múltiples aplicacions comercials utilitzen aquest format de vídeo, YouTube, Google Video, Reuters i MySpace en són alguns exemples.

Els clients web d'aquest Treball reben els missatges de senyalització del seu corresponent B2BUA. Utilitzen la informació proporcionada per subscriure's als fluxos que corresponen als usuaris de la trucada i publiquen el seu propi flux amb el seu nom com a identificador.

[1] Flash Video (Consultar els ACRÒNIMS)

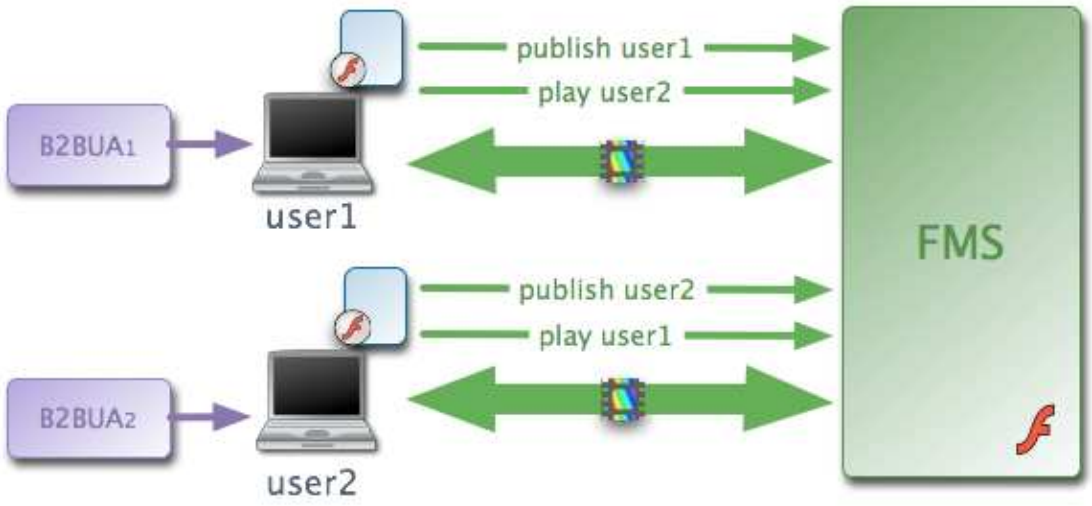


Fig. 2.8 Arquitectura de publicació i subscripció a fluxos

La **¡Error! No se encuentra el origen de la referencia.** mostra un exemple d'aquest procés amb dos usuaris, "user1" i "user2". Cadascun rep la informació sobre l'altre a través dels respectius B2BUA's. L'user1 publica el seu flux d'àudio i vídeo amb l'identificador "user1" i es subscriu al flux de l'altre usuari amb identificador "user2". L'user2 segueix el mateix procediment. El Flash Media Server s'encarrega d'obtenir els fluxos i enviar-los als usuaris que correspongui.

2.5. Diagrames d'interacció entre els diferents mòduls

Aquest apartat pretén concretar en el funcionament general de l'aplicació. Per això es mostren una sèrie de diagrames d'interacció que fan més entenedor el mecanisme utilitzat per la comunicació entre els diferents mòduls del sistema.[1]

[1] Per entendre millor els missatges XML que s'envien el WiCat Client i Server consultar l'ANNEX D DISSENY DEL PROTOCOL DE COMUNICACIÓ.

2.5.1. Inicialització

El següent diagrama mostra els passos que segueix l'aplicació després que l'usuari introdueixi el seu nom i contrasenya.

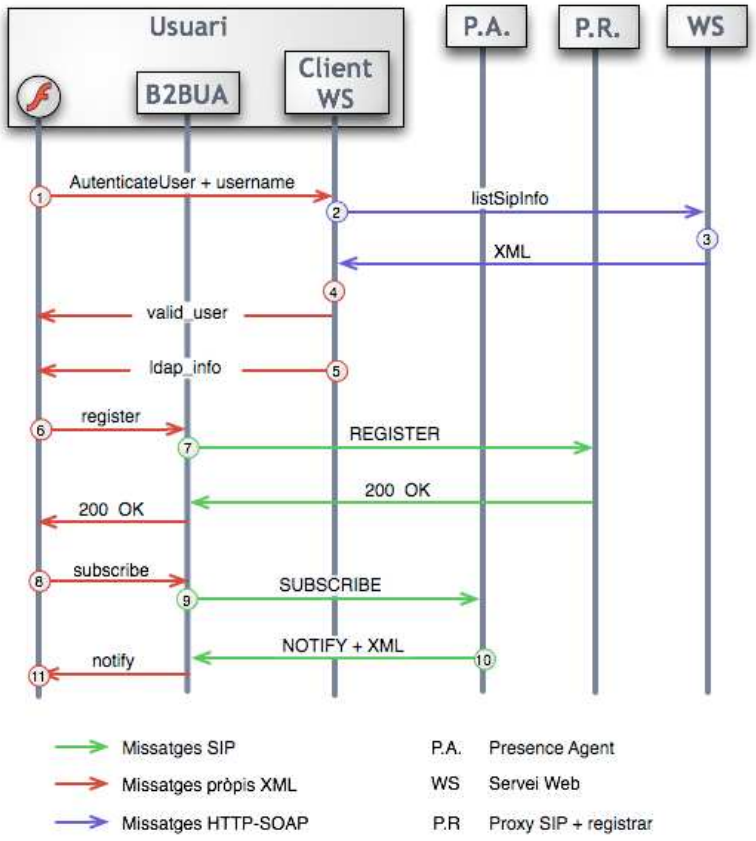


Fig. 2.9 Inicialització d'un nou usuari

1.- El client Flash envia un missatge XML de control (DataMessage) de tipus *AutenticateUser*. Aquest XML incorpora dos tags amb el nom de l'usuari i la contrasenya.

2.- El client del servei web fa una crida HTTP-SOAP a la funció *listSipInfo* del servei web.

3.- Després de consultar el directori LDAP, el servei web retorna la llista de persones amb la seva corresponent SIP URI. Per més informació veure apartat 3.1.2.

4.- La llista que es rep del servei web ha de contenir una entrada amb el nom de l'usuari. Si la conté, s'enviarà a l'aplicació Flash un missatge XML de tipus *AutenticateUser*, amb un tag <*valid_user*> amb el valor *true*. Després d'això es passa al pas 5.

Si la llista no conté una entrada que correspongui amb l'usuari, s'enviarà un XML com l'anterior però amb el tag <*valid_user*> a *false*. L'aplicació quedarà a l'espera d'un nom d'usuari vàlid.

5.- El client del servei web enviarà la llista *listSipInfo* rebuda cap al client Flash. Aquesta llista anirà encapsulada dins d'un *DataMessage* on el tag <*message_type*> tindrà el valor *listSipInfo*.

6.- Després de rebre la llista d'entrades LDAP, el client emmagatzema la informació dels usuaris i envia un missatge XML al B2BUA de tipus *register*. En aquest missatge especifica el nom d'usuari i la SIP URI obtinguda del LDAP.

7.- El B2BUA inicia la pila de SIP amb el nom de l'usuari rebut. Després envia un REGISTER al proxy SIP registrar, segons la SIP URI de l'usuari.

8.- Quan el client Flash rep la resposta 200 OK del seu B2BUA, es subscriu al event de presència. Amb un XML *SipMessage* de tipus *subscribe*.

9.- El B2BUA envia un SUBSCRIBE al Presence Agent amb *event* = *"presence"*[1].

10.- El Presence Agent subscriu al usuari amb *status online* i notifica tots els seus subscriptors amb un NOTIFY que conté la llista dels usuaris[2].

11.- Quan el client Flash rep la informació de presència actualitza el camp *status* de cada usuari emmagatzemat. Seguidament mostra la llista d'usuaris amb els corresponents indicadors de presència (color verd si l'usuari està connectat i vermell si està desconnectat).

[1] Per més informació sobre el mecanisme de subscripció veure l'apartat 3.2.2.

[2] Per més informació sobre el mecanisme de notificació veure l'apartat 3.2.4.

2.5.2. Videoconferència senzilla

El següent diagrama correspon a una trucada simple. L'usuari1 truca a l'usuari2. Es pot veure l'intercanvi de missatges des que l'usuari 1 prem el botó de trucar fins que s'acaba la videoconferència.[1]

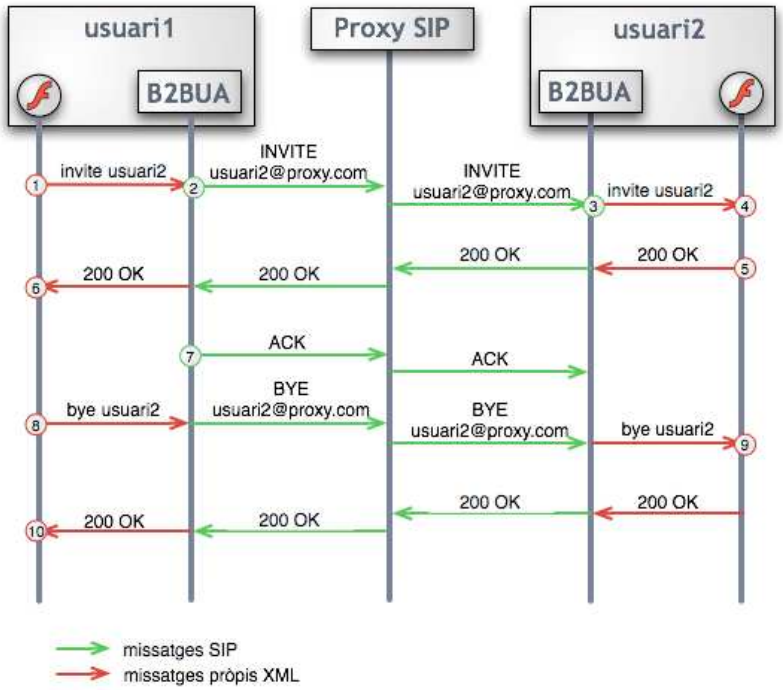


Fig. 2.10 Videoconferència senzilla

1.- L'usuari1 selecciona un usuari connectat de la llista de presència i prem el botó per trucar. L'aplicació Flash envia un missatge XML de tipus *invite*.

2.- El B2BUA rep l'XML del client Flash i envia un INVITE a la SIP URI indicada en el missatge XML (usuari2@proxy.com).

3.- El proxy SIP reenvia l'INVITE al B2BUA de l'usuari2 i aquest envia un XML de tipus *invite* al client Flash.

4.- El client Flash del usuari2 processa l'XML i mostra una finestra d'alerta amb la informació de la trucada (usuari que la inicia).

5.- Si l'usuari2 accepta la trucada, el client Flash envia un XML amb una resposta 200 OK per l'usuari1. Seguidament comença a transmetre el flux d'àudio i vídeo cap al Flash Media Server.[2]

---

[1] En aquest diagrama i els que es troben a continuació s'ha omès la comunicació amb el Flash Media Server per simplificar l'esquema.

[2] Per més informació sobre la comunicació amb el Flash Media Server, veure l'annex B.2.

6.- Quan l'usuari1 rep el 200 OK de l'usuari2, el seu client Flash comença la comunicació amb el Flash Media Server i s'inicia la videoconferència.

7.- Un cop retransmès el 200 OK cap a l'aplicació Flash, el B2BUA envia l'ACK corresponent. Aquest ACK no arriba als extrems de la comunicació, es queda en el B2BUA.

8.- L'usuari1 prem el botó de penjar. L'aplicació Flash crea un XML de tipus *bye* per l'usuari2.

9.- L'usuari2 rep el *bye* i acaba la trucada. Tanca la connexió amb el Flash Media Server i respon amb un 200 OK.

10.- Quan l'usuari1 rep el 200 OK de l'usuari2, tanca la connexió amb el Flash Media Server i la videoconferència s'acaba.

2.5.3. Multi conferència senzilla

El següent diagrama mostra la seqüència de missatges, tant del protocol propi com SIP, per realitzar una multi conferència. Per multi conferència entenem una videoconferència en la qual hi participen més de dos participants.

L'Usuari Master és qui selecciona els usuaris de la llista de presència i clica el botó de trucar. L'Usuari Simple és un dels usuaris trucats.

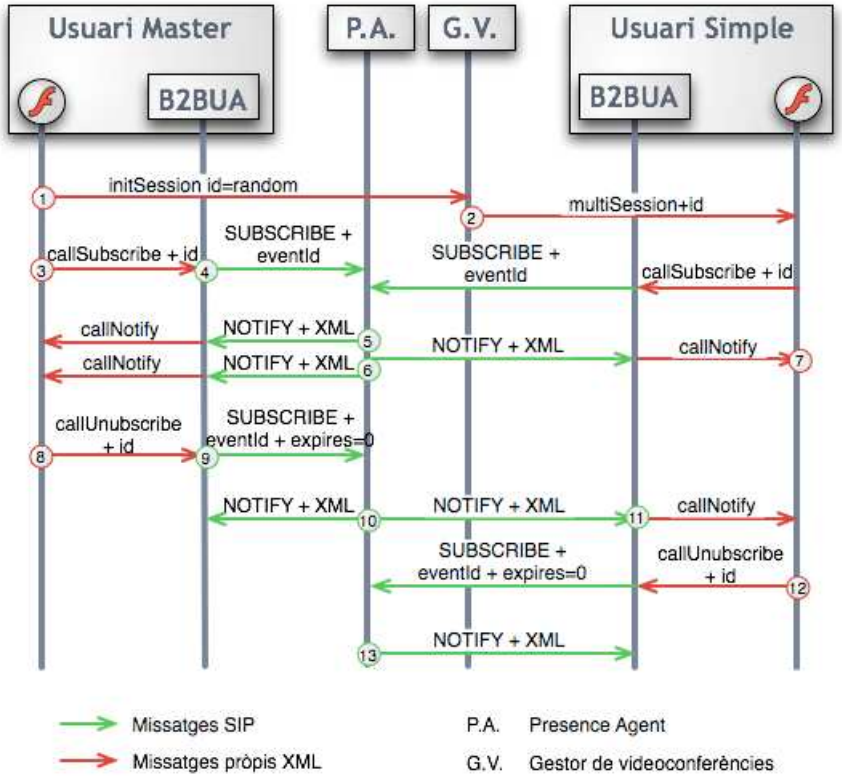


Fig. 2.11 Multi conferència senzilla

1.- L'Usuari Master selecciona més d'un usuari connectat i clica el botó de trucar. L'aplicació Flash envia un XML de tipus *initSession*. Aquest missatge conté el nom de l'usuari master, el nom dels usuaris trucats i un identificador de trucada que es genera aleatòriament.

2.- El gestor de videoconferències rep el missatge de l'usuari Master i envia a la resta d'usuaris implicats en la trucada un missatge XML *multiSession* on afegeix la informació sobre la multi conferència.

3.- L'usuari Simple rep un avís de multi conferència perquè s'hi uneixi. Si accepta la invitació, l'aplicació Flash enviarà un XML de tipus *callSubscribe* on s'especifica l'identificador de la multi conferència. L'usuari Master després d'enviar el missatge *multiSession* també es subscriu a la trucada.

4.- Els B2BUA dels corresponents usuaris, reben el missatge *callSubscribe* i envien un SUBSCRIBE al Presence Agent. A aquest missatge SIP li afegeixen una capçalera *event = "presence"* i un camp *eventId* que correspon amb l'identificador de la videoconferència.

5.- El Presence Agent rep el primer SUBSCRIBE (el de l'usuari Master) i li envia un NOTIFY amb un XML que conté els noms dels usuaris subscrits a la trucada. En aquest cas només conté el nom de l'usuari Master.

6.- Quan el Presence Agent rep un altre SUBSCRIBE (el de l'usuari Simple), actualitza la llista d'usuaris implicats en la videoconferència i envia un NOTIFY amb aquesta llista a tots els subscriptors.

7.- Els clients Flash reben un *callNotify* dels corresponents B2BUA amb el mateix XML que envia el Presence Agent en el NOTIFY. D'aquest XML extreuen els noms dels usuaris i es subscriuen al flux d'àudio i vídeo de cadascun a través del Flash Media Server.

En aquest punt la multi conferència està iniciada. A partir d'aquí, cada cop que un client Flash rebi un *callNotify* actualitzarà la llista d'usuaris de la videoconferència i es subscriurà als fluxos que correspongui.

8.- Quan un usuari vol abandonar la videoconferència clica el botó de penjar. L'aplicació Flash envia un XML de tipus *callUnsubsribe* i es desconnecta del Flash Media Server.

9.- El B2BUA de l'usuari que abandona enviarà un SUBSCRIBE al Presence Agent. Aquest SUBSCRIBE tindrà les mateixes característiques que l'enviat al inici, amb la diferència que la capçalera *expires* tindrà valor zero.

10.- Quan el Presence Agent rep el SUBSCRIBE amb *expires = 0* elimina la l'usuari de la llista i notifica a tots els subscriptors amb l'XML actualitzat.

11.- El B2BUA de l'usuari Master no reenvia el NOTIFY que rep del Presence Agent ja que l'usuari ha sortit de la multi conferència i no necessita la informació. En canvi, el B2BUA de l'usuari Simple reenviarà l'XML rebut mitjançant un missatge *callNotify*.

12.- Quan el client Flash rep un *callNotify* comprova en la llista rebuda si hi ha algun usuari subscrit a la videoconferència a part d'ell mateix. Si no n'hi ha cap altre, automàticament tancarà la connexió amb el Flash Media Server i enviarà un missatge *callUnsubscribe* cap al B2BUA.

13.- El Presence Agent actualitzarà la llista d'usuaris de la videoconferència i enviarà l'últim NOTIFY pel B2BUA de l'usuari Simple. Com que no queden usuaris subscrits eliminarà la subscripció de la videoconferència.

2.5.4. Multi conferència programada

El següent diagrama mostra el comportament dels diferents elements implicats en l'establiment i acabament de les videoconferències programades. L'usuari Master se suposa que ha programat una multi conferència amb varis usuaris, entre ells l'usuari Simple.

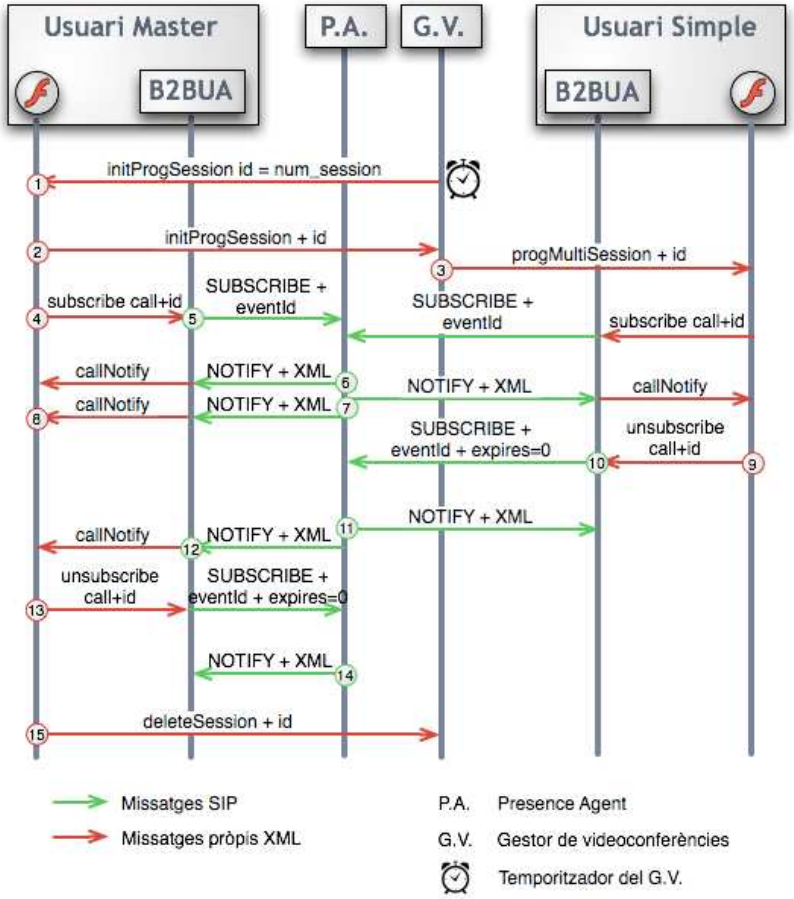


Fig. 2.12 Multi conferència programada

1.- Quan arriba l'hora programada per iniciar la multi conferència, el Gestor de Videoconferències envia a l'usuari Master un XML *initProgSession* amb la informació sobre la trucada: usuaris implicats, descripció i identificador (l'identificador l'assigna el Gestor de Videoconferències quan es programa).

2.- L'aplicació Flash mostra a l'usuari Master una finestra d'avís de multi conferència. En aquesta finestra pot escollir iniciar la multi conferència o no. Si escull iniciar-la l'aplicació Flash envia un XML *initProgSession* cap al Gestor de Videoconferències amb la mateixa informació que contenia el rebut.

3.- Quan rep la confirmació de l'usuari Master, el Gestor de Videoconferències s'encarrega d'avisar a la resta d'usuaris implicats mitjançant un missatge

progMultiSession. A aquests usuaris se'ls mostra una finestra d'alerta on poden escollir unir-se a la videoconferència.

A partir d'aquest punt l'intercanvi de missatges és el mateix que per les multi conferències simples. Per recordar-ho es poden consultar els punts 3 a 14 de l'esquema de multi conferència (**¡Error! No se encuentra el origen de la referencia.**) que corresponen amb els punts 4 a 14 d'aquest esquema (**¡Error! No se encuentra el origen de la referencia.**). L'única diferència és que en aquest cas el primer en abandonar la trucada és l'usuari Simple, l'usuari Master l'abandona quan no hi ha més usuaris subscrits.

15.- En aquest punt l'usuari Master ha enviat un missatge *callUnsubscribe* per abandonar la multi conferència. Perquè el Gestor de Videoconferències s'adoni que la trucada s'ha acabat, l'aplicació Flash de l'usuari Master envia un missatge *deleteSession* indicant l'identificador de la multi conferència.

2.5.4.1. Ingrés d'usuaris en multi conferències programades

L'esquema anterior (**¡Error! No se encuentra el origen de la referencia.**) correspon amb la manera simple d'unir-se a una multi conferència programada. S'ha suposat que els usuaris implicats en la trucada estan connectats a l'aplicació quan arriba l'hora d'iniciar-la. Què passa si un dels usuaris convidats a la multi conferència es connecta quan aquesta ja ha començat? El següent esquema ho il·lustra.

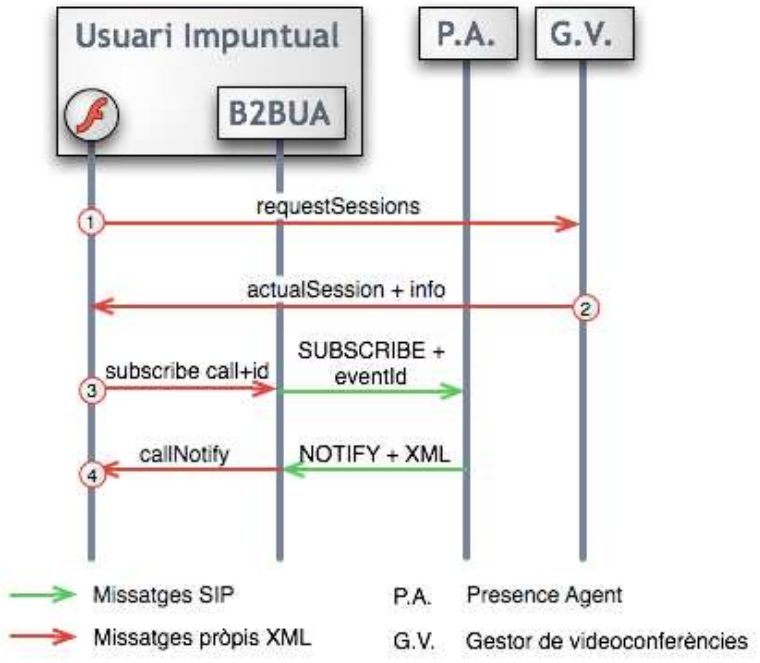


Fig. 2.13 Ingrés d'un usuari en una multi conferència programada

1.- L'usuari es connecta a l'aplicació quan hi ha una multi conferència en curs a la que està convidat. Si clica el botó "consultar videoconferències programades", l'aplicació Flash enviarà un missatge *requestSessions* al Gestor de Videoconferències.

2.- El Gestor comprovarà si hi ha alguna multi conferència en curs, si és així i l'usuari hi apareix com a convidat, retornarà un missatge *actualSession* on indicarà la informació sobre la videoconferència.

3.- L'aplicació Flash mostrarà una finestra d'alerta a l'usuari informant de la videoconferència. Si escull unir-s'hi s'enviarà un *subscribe* a la trucada igual que passava en els esquemes anteriors.

4.- Tant l'usuari Impuntual com tots els presents en la videoconferència, rebran un *callNotify* amb la llista d'usuaris actualitzada. L'usuari Impuntual publicarà el seu flux d'àudio i vídeo mitjançant el Flash Media Server i tots els demés s'hi subscriuran. D'aquesta manera l'usuari Impuntual ja es pot comunicar amb la resta.

Aquest esquema és possible sempre que el Gestor de Videoconferències tingui constància que la multi conferència s'està realitzant en aquell moment. Com s'ha vist en l'esquema anterior (**¡Error! No se encuentra el origen de la referencia.**), el Gestor elimina la informació sobre la multi conferència quan l'usuari Master envia un missatge *deleteSession*. Això vol dir que cap usuari impuntual s'hi podrà unir si l'usuari Master ha abandonat la trucada. Encara que altres usuaris continuïn comunicant-se, el Gestor de Videoconferències considera la videoconferència acabada quan l'usuari Master en surt.

2.5.5. Desconnexió de l'usuari

Hi ha dos casos en els quals l'usuari es desconnecta de l'aplicació. El primer cas es produeix quan l'usuari clica el botó de sortir en la interfície web. El segon, quan es produeix algun error en la comunicació entre el client i el WiCat Server o quan l'usuari tanca el navegador. En el primer cas el WiCat Client envia un missatge XML de tipus *exit* i tanca la connexió amb el servidor. En el segon, el servidor detecta la desconnexió de l'usuari. En tots dos casos el servidor s'ocuparà d'actualitzar l'estat d'aquest usuari. La desconnexió es pot produir en varies situacions:

L'usuari no està realitzant cap videoconferència.

El B2BUA enviarà un SUBSCRIBE amb *expires = 0* i *event = "presence"* cap al Presence Agent per eliminar la subscripció de l'usuari a la presència. Quan rebi el NOTIFY del Presence Agent, enviarà un REGISTER amb *expires = 0* cap al proxy Registrar perquè aquest l'elimini de la base de dades. Quan rebi el 200 OK del proxy el WiCat Server s'encarregarà de finalitzar la pila SIP i d'alliberar els recursos d'aquest usuari.

L'usuari està realitzant una videoconferència senzilla, amb un altre usuari.

El B2BUA enviarà un missatge BYE cap a l'altre usuari. Quan rebi el 200 OK realitzarà les mateixes accions que en el cas anterior.

L'usuari és un convidat a una multi conferència senzilla o programada, o és Master d'una multi conferència senzilla.

El B2BUA enviarà un SUBSCRIBE amb *expires = 0* i *event = "presence"* al Presence Agent. Aquest eliminarà la subscripció de l'usuari a l'event *presence* i a més, com que l'usuari també forma part d'una multi conferència, eliminarà la subscripció a ella. Després de rebre el NOTIFY del Presence Agent finalitzarà igual que un usuari normal.

L'usuari és Master d'una multi conferència programada que està en curs.

En aquest cas el gestor de videoconferències eliminarà la videoconferència de la llista perquè cap altre usuari s'hi pugui subscriure. La resta de passos són els mateixos que en el cas anterior.

CAPÍTOL 3. IMPLEMENTACIÓ

En aquest capítol es descriu la implementació feta de cada mòdul del sistema, seguint la mateixa classificació que en el capítol anterior. A l'última secció del capítol es mostra l'escenari de probes utilitzat per la realització del treball i el resultat de les probes fetes amb l'aplicació en funcionament.

3.1. Gestió d'usuaris i equips de videoconferències

En les seccions següents es dóna una visió més detallada de les tecnologies utilitzades per la gestió dels usuaris i d'equips de videoconferència i de la implementació d'aquestes tecnologies en l'aplicació WiCat.

Per indexar la informació dels usuaris d'una organització s'ha utilitzat un directori basat en el protocol LDAP.

3.1.1. LDAP

L'elecció d'aquest sistema s'ha fet, en part, gràcies a la facilitat que proporciona a l'hora d'emmagatzemar informació d'usuaris d'una organització i de vincular-los a terminals de videoconferència. Això és possible gràcies a l'estàndard de la ITU-T H.350. Seguidament es proporciona una introducció al sistema de directori LDAP, i es relaciona amb l'H350.

3.1.1.1. LDAP i H350

La informació d'un LDAP està organitzada en forma d'arbre, cada fulla d'aquest arbre és una entrada, l'entrada de nivell més alt és l'entrada *root.* Cada entrada inclou un *distinguished name* (DN) i varies parelles d'atribut/valor. El DN és el nom de l'entrada i ha de ser únic. Aquest mostra la relació entre l'entrada i la resta de l'arbre LDAP de manera similar a com la ruta completa d'un fitxer mostra la serva relació amb la resta de fitxers del sistema. Tot seguit es pot observar un exemple d'un DN:

uid=Joan,ou=persona,o=i2cat

La primera part del DN s'anomena *relative distinguished name* (RDN) i està composada per una parella d'atribut/valor. En l'exemple el RDN és *uid=Joan.*
En la Taula 3.1 es poden veure els atributs utilitzats en una entrada personal del directori LDAP utilitzat en aquest Treball.

Taula 3.1 Atributs d'una entrada LDAP

Atribut	Descripció
o	Organització
ou	Unitat organitzacional
cn	Nom
sn	Cognom
uid	ID de l'usuari
dn	*distinguished name*
mail	Adreça email
commURI	Adreça LDAP que apunta al terminal de videoconferència

L'H350 defineix una sèrie de classes i d'atributs relacionats amb protocols multimèdia. La classe principal s'anomena *commObject* (communications Object class). Aquest objecte es pot utilitzar per representar terminals h.323, user agents SIP, etc. Permet representar un terminal multimèdia amb una entrada de directori LDAP.

Els atributs *commOwner* i *commUri* s'utilitzen per relacionar una entrada LDAP personal amb la informació específica del seu terminal de veu sobre IP i alhora relacionar el terminal amb el seu propietari.

Afegint un "punter" *commURI* en l'entrada LDAP d'una persona, aquesta persona queda associada amb el *commObject* en qüestió.

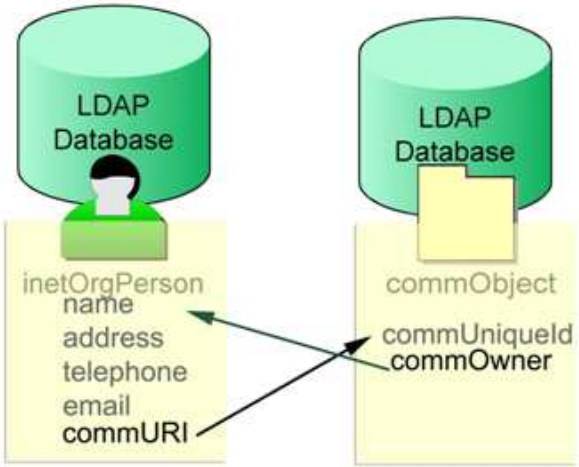


Fig. 3.1 Associació entre persones i terminals de videoconferència

L'entrada *commObject* conté un punter anomenat *commOwner* que apunta a la persona associada amb aquest *commObject*. D'aquesta manera persona i terminal queden relacionats.

Per afegir suport d'H.350 al directori LDAP d'una organització només cal afegir l'objecte *commUri* en les entrades personals. Els *commObject* poden ser instanciats en el mateix LDAP o poden residir en directoris separats.

3.1.1.2. LDAP i JNDI

Per accedir al servei de directori LDAP s'ha utilitzat una API de Java anomenada JNDI[1]. Aquesta API proporciona mètodes per consultar i modificar la informació continguda en un arbre LDAP. Per exemple, és possible buscar una entrada pels seus atributs o associar nous atributs a una entrada. En la Taula 3.2 es poden observar algunes de les operacions que es poden realitzar a un LDAP i els seus equivalents en JNDI.

Taula 3.2 Operacions LDAP i equivalents JNDI

Operació	Finalitat	Equivalent JNDI
Search	Busca en el directori per trobar entrades	DirContext.search()
Add	Afegeix una nova entrada en el directori	DirContext.bind(), DirContext.createSubcontext()
Modify	Modifica una entrada del directori en particular	DirContext.modifyAttributes()
Delete	Elimina una entrada del directori	DirContext.unbind(), DirContext.destroySubcontext()
Rename	Canvia el nom o modifica el DN	DirContext.rename()
Bind	Inicia una sessió amb un servidor LDAP	new InitialDirContext()
Unbind	Acaba una sessió amb un servidor LDAP	DirContext.close()

3.1.2. Serveis Web

Els serveis web comunament fan servir el protocol HTTP-SOAP basat en XML i defineixen les seves interfícies en un arxiu WSDL. Per desplegar el servei web utilitzat en aquest treball s'ha utilitzat el Framework Apache Axis.[2]

[1] Java Naming and Directory Interface (Veure els ACRÒNIMS)
[2] Per més informació sobre el desplegament de serveis web amb l'entorn d'Apache Axis veure ANNEX C

El servei web desenvolupat està destinat a pertànyer a una organització que disposi d'un directori LDAP del qual no es vulgui donar accés des de l'exterior per seguretat, però amb la necessitat de fer-hi consultes des d'Internet.

Per complir aquest propòsit el servei web disposa de varis mètodes accessibles des d'Internet que proporcionen informació sobre el directori LDAP de l'organització a la qual pertanyi el servei web. La classe que conté aquests mètodes s'anomena Sldap, a continuació es mostra el diagrama de classe.

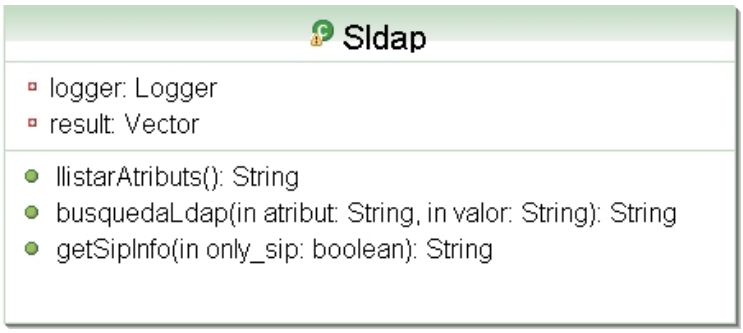


Fig. 3.2 Diagrama de la classe Sldap

Seguidament s'expliquen les operacions que realitza cada mètode i la informació que retorna.

- **busquedaLdap:** Aquest mètode troba les entrades LDAP que contenen l'atribut i el valor especificats. És possible especificar un valor igual a "*" per trobar totes les entrades que continguin l'atribut sense tenir en compte el valor. Les entrades trobades s'emmagatzemen en objectes LdapResult i es converteixen a un XML que segueix aquest format:

```
<?xml version='1.0' encoding='iso-8859-1'?>
<list_ldapResult>
  <status>aquest tag indica si la cerca ha tingut èxit o no
  </status>
  <ldapResult>
     <name>common name</name>
     <attribute id="id_atribut" >valor_atribut</attribute>
  </ldapResult>
  <ldapResult>
   …
  </ldapResult>
  …
</list_ldapResult>
```

- **llistarAtributs:** Aquest mètode retorna tots els atributs del directori LDAP ignorant les entrades a les quals pertanyen i el seu valor. Retorna un XML amb aquest format:

```
<?xml version='1.0' encoding='iso-8859-1'?>
<list_attributes>
  <status>aquest tag indica si la cerca ha tingut èxit o no
  </status>
  <attribute>id_atribut</attribute>
  <attribute>id_atribut</attribute>
   …
</list_attributes>
```

- **getSipInfo:** Aquest mètode rep una variable booleana per paràmetre. Aquesta variable indica si la cerca s'ha de fer només per les entrades que contenen informació SIP o per totes les entrades. Tant si com no, les entrades que contenen informació SIP es relacionen amb l'entrada corresponent al seu terminal i se'ls afegeix un atribut amb la seva SIP URI com a valor. El format de l'XML de retorn és el següent:
-

```
<?xml version='1.0' encoding='iso-8859-1'?>
<list_sipInfo>
  <status>aquest tag indica si la cerca ha tingut èxit o no
  </status>
  <ldapResult>
    <name>common name</name>
    <attribute id="id_atribut" >valor_atribut</attribute >
    <sipUri>aquest tag apareix si l'entrada conté informació
            sip </sipUri>
  </ldapResult>
  <ldapResult> … </ldapResult>
   …
</list_sipInfo >
```

Segons el mètode consultat es retorna l'XML corresponent mitjançant HTTP-SOAP cap al client del servei web.

3.1.3. Client del Servei Web

Per cada client connectat al WiCat Server es crea una instància del client del servei web. Aquest client disposa dels mateixos mètodes que el servei web i permet obtenir la llista d'usuaris de la organització i la informació sobre cadascun d'ells.

El mètode consultat per obtenir la llista d'usuaris és el *getSipInfo*. Es consulta aquest mètode perquè és necessari obtenir la SIP URI dels usuaris per establir-hi una videoconferència. El resultat que s'obté d'aquesta consulta és l'XML mostrat en l'apartat anterior.

El client del servei web disposa de la classe LdapResult i AttributeLdap que també es troben en el servei web. La classe LdapResult emmagatzema una entrada LDAP. Cada objecte LdapResult es composa de varis objectes AtributeLdap que emmagatzemen l'identificador d'un atribut i el seu valor.

3.2. Senyalització i gestió de videoconferències

Per poder integrar un protocol de senyalització de VoIP com SIP s'ha hagut de dividir l'aplicació web en tres parts: una que pugui enviar i rebre vídeo i que, a més sigui accessible des de qualsevol ordinador amb Internet, l'altra que pugui mantenir la senyalització de les videoconferències i una última que proporcioni la informació necessària per contactar amb els usuaris.

El WiCat Server manté la comunicació amb els clients Flash i s'ocupa de la part de senyalització amb una pila SIP. Tot seguit s'explicarà la implementació d'aquest servidor, amb l'ajuda d'un diagrama.

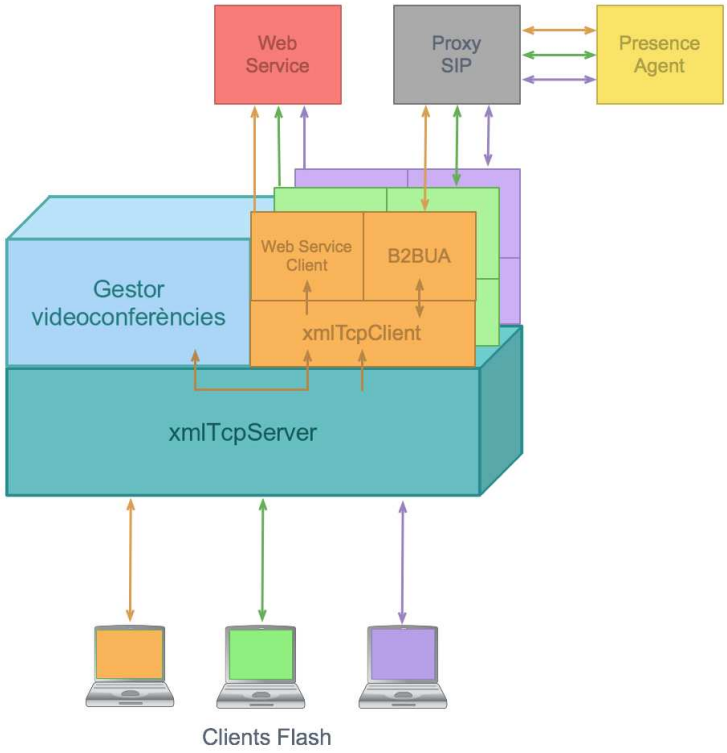


Fig. 3.3 Detall de l'arquitectura del WiCat Server

En la Fig. 3.3 s'observa l'arquitectura interna del WiCat Server i la comunicació amb la resta de mòduls. El servidor disposa d'un *ServerSocket* esperant connexions dels clients Flash (xmlTcpServer). Aquest servidor atén a tots els clients d'una manera concurrent.

Cada client Flash (WiCat Client) connectat a l'aplicació disposa d'un mòdul dedicat a la comunicació amb els elements externs (servei web, proxy SIP i Presence Agent), i també amb el Gestor de Videoconferències situat en el mateix WiCat Server.

La classe que s'encarrega d'atendre cada client Flash s'anomena XMLTcpClient. Aquesta classe fa de passarel·la cap al B2BUA SIP i cap al client del servei web.

En els subapartats següents s'explica la implementació dels elements de l'arquitectura encarregats de la senyalització i la gestió de videoconferències.

3.2.1. Client Web

Com ja s'ha comentat, el WiCat Client no disposa d'una pila SIP per encarregar-se de la senyalització. Aquesta senyalització es realitza en la part del WiCat Server mitjançant el B2BUA. Encara que el client Flash no sigui un user agent SIP, ha de rebre els missatges de senyalització, com INVITE's, NOTIFY's, etc. Per això s'ha dissenyat un sistema de comunicació amb el servidor basat en XML.

3.2.1.1. XMLSocket

Aquesta classe de Flash permetrà al client Flash mantenir una comunicació continua amb el WiCat Server.

L'XMLSocket implementa un *socket* client per comunicar el Flash Player amb un servidor extern desenvolupat en qualsevol altre llenguatge de programació. Els missatges s'envien sobre una connexió TCP full-duplex. Cada missatge és un XML complet acabat amb un '\0'.

La connexió amb el servidor s'estableix al iniciar l'aplicació i es manté durant tota la sessió. El *socket* es tanca quan es prem el botó "sortir" o quan es tanca el navegador. Aquest tipus de comunicació és adequada per aquest treball, ja que permet la recepció de missatges del servidor quan es produeix algun event, no només sota demanda del client com es produiria en una connexió basada en HTTP.

3.2.1.2. Missatges XML

S'han definit una sèrie de missatges XML per comunicar client i servidor. Per una banda hi ha els missatges SIP, aquests segueixen la mateixa estructura, tot seguit se'n pot veure un exemple, el cas del missatge INVITE.

```
<?xml version='1.0' encoding='iso-8859-1'?>
<SipMessage>
     <message_type>invite</message_type>
        <sender_client>nom_client_local</sender_client>
     <receiver_client>nom_client_receptor</receiver_client>
     <receiver_sipUri>client_receptor@proxy.com
     </receiver_sipUri>
</SipMessage>
```

Els missatges propis de tipus SIP tenen en comú el tag principal <*SipMessage*>. Per especificar el tipus de missatge SIP representen, tots incorporen el tag <message_type>.

Per l'altra banda són necessaris també missatges de control, propis de l'aplicació, utilitzats per la gestió de videoconferències i d'usuaris. Un exemple és el missatge que rep el client Flash amb la informació de les multi conferències programades.

```
<?xml version='1.0' encoding='iso-8859-1'?>
<DataMessage>
     <message_type>session_entries</message_type>
        <session>
                <sessionId>405</sessionId>
                <sessionMaster>nom_user_master</sessionMaster>
                 <user>nom_user1</user>
                 ...
                 <user>nom_userN</user>
                <year>2007</year>
                <month>1</month>
                <day>20</day>
                <hour>15</hour>
                <minute>45</minute>
                <description>descripció_videoconferència
                </description>
        </session>
</DataMessage>
```

Aquest tipus de missatges tenen una estructura comuna. El tag principal és <*DataMessage*>,el format del XML que hi ha en l'interior s'especifica en el tag <*message_type*>. Segons el valor del tag <*message_type*>, tant el client Flash com el servidor Java, sabran com processar la resta del missatge.

3.2.1.3. Comunicació local

La web esta formada per varis arxius SWF, cadascun es carrega per decisió de l'usuari o per la recepció de certs missatges del servidor. És necessari que amb un sol XMLSocket tots els arxius que conformen la web puguin enviar i rebre dades del servidor.

S'ha utilitzat la classe LocalConnection de Flash, que permet comunicar dos pel·lícules de Flash (arxius SWF) executant-se en la mateixa màquina. Per exemple, pot comunicar una pel·lícula que s'executa en el navegador Internet Explorer i una altra que s'executa en el Firefox. En aquest treball s'utilitza el LocalConnection per comunicar dos pel·lícules de Flash executant-se en el mateix navegador.

La classe LocalConnection estableix una connexió punt a punt i d'un sol sentit, si volem una comunicació *full-duplex* seran necessaris dos objectes LocalConnection. El diagrama següent mostra el funcionament d'aquesta connexió.

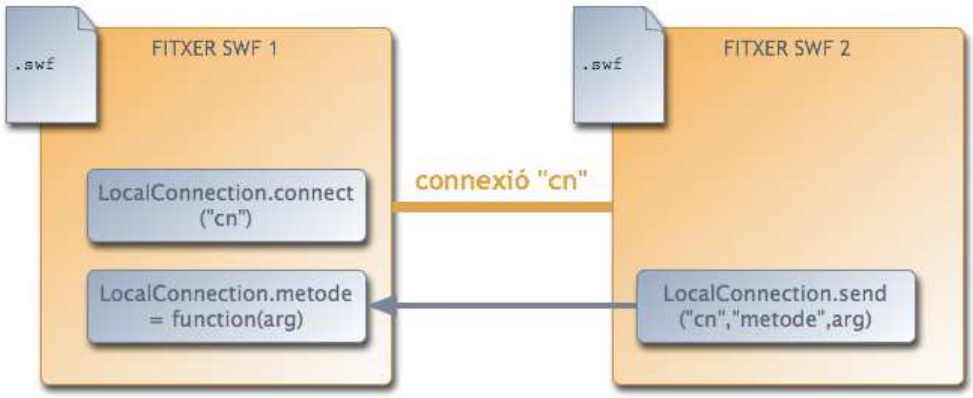


Fig. 3.4 Funcionament de l'objecte LocalConnection

El fitxer SWF 1 prepara l'objecte LocalConnection per rebre crides des del mètode LocalConnection.send() a la connexió "cn". Declara la funció "metode" que es cridarà des del fitxer SWF 2.

El fitxer SWF 2 invoca a la funció "metode" de la connexió "cn" i li passa per paràmetre "arg".

3.2.2. B2BUA

Com s'ha explicat en el capítol anterior, aquest element s'encarrega de la senyalització SIP dels usuaris. Existeix un B2BUA per cada client web connectat a l'aplicació.

El B2BUA ha estat desenvolupat gràcies a la API de Java JAIN SIP. Aquest element implementa els següents missatges SIP:

REGISTER: Aquest missatge l'envia per registrar l'usuari a la base de dades del proxy Registrar.

INVITE: Envia o rep un INVITE per iniciar una trucada.

BYE: L'envia i el rep per acabar una trucada.

SUBSCRIBE: Aquest missatge l'envia cap al Presence Agent per subscrire's a un cert event.

NOTIFY: El rep del Presence Agent amb informació sobre l'event al que està subscrit.

El missatge SUBSCRIBE té doble funcionalitat. Per una banda serveix als usuaris per poder rebre notificacions sobre l'estat dels altres usuaris (*online* o *offline*), i per l'altra gestiona la presència d'usuaris en les sales de multi conferència. La implementació del mecanisme de subscripció s'ha realitzat seguint l'RFC 3265 [4] Seguidament s'expliquen els dos tipus de missatges SUBSCRIBE:

Presència dels usuaris:

Després de fer un REGISTER els B2BUA's dels usuaris envien un SUBSCRIBE al Presence Agent. Aquest SUBSCRIBE conté una capçalera anomenada "Event" on s'indica l'event al qual s'estan subscrivint. En aquest cas l'event utilitzat és "presence".

També s'afegeix una capçalera "Expires" on s'indica el temps que durarà la subscripció al event. Abans que caduqui la subscripció el client haurà d'enviar un altre SUBSCRIBE o el Presence Agent eliminarà la seva subscripció. Per assegurar que no caduca la subscripció de l'usuari, després d'enviar el primer SUBSCRIBE es comprova cada segon si en queden menys de trenta de subscripció. Si es dóna el cas s'envia un altre SUBSCRIBE amb la capçalera "Expires" actualitzada.

Quan l'usuari surt de l'aplicació s'envia un SUBSCRIBE amb la capçalera "Expires" de valor zero. Quan el Presence Agent rep un SUBSCRIBE d'aquestes característiques elimina la subscripció de l'usuari en qüestió.

Presència dels usuaris en "sales" de videoconferència:

Quan un usuari vol entrar en una multi conferència envia un SUBSCRIBE al PresenceAgent variant la capçalera "Event". L'event al qual es subscriu en aquest cas és "call_presence" i afegeix un paràmetre opcional "EventId" que correspon amb l'identificador de la videoconferència.

Aquest tipus de SUBSCRIBE també conté una capçalera "Expires" que indica el temps que durarà la subscripció a la multi conferència, com en el cas anterior s'inicia un *Timer* que controla aquest temps.

Per sortir de la multi conferència l'usuari enviarà un SUBSCRIBE amb l'"Expires" de valor zero.

3.2.2.1. Implementació del B2BUA

Les classes utilitzades per la recepció i enviament de missatges SIP estan realitzades amb la API JAIN SIP, concretament la implementació feta per NIST ja que és lliure i oberta. La llibreria JAIN SIP ofereix una pila SIP que permet el desenvolupament de tot tipus d'aplicacions que utilitzin l'estàndard SIP. L'estructura utilitzada consta de cinc classes:

- **SipActor**: Aquesta classe proporciona els mètodes i variables necessàris per crear peticions SIP *request*.

- **MySip**: Extèn de SipActor, permetent utilitzar totes les variables i mètodes d'aquesta. A més implementa la interfície SipListener, això li permet rebre missatges SIP. També s'encarrega de generar *responses* als missatges rebuts.

- **SipClient**: Interfície que defineix els mètodes necessaris pel control del mòdul de senyalització des d'altres classes.

- **SipClientImpl**: Aquesta classe és la implementació de SipClient. Fa de passarel·la entre el xmlTcpClient (que s'encarrega d'atendre a un WiCat Client) i la pila SIP. Guarda l'estat de les trucades, evitant el mal us dels missatges SIP que pot originar errors.

- **Subscriber**: Aquest tipus d'objecte s'utilitza en la classe SipActor pel control de les subscripcions de l'usuari, tant per la presència d'usuaris com per la presència en sales de videoconferència. S'encarrega de controlar les subscripcions, reenviar SUBSCRIBE's en cas que caduquin,etc.

3.2.2.2. Justificació de l'ús de sales de videoconferència

Al principi d'aquest apartat s'ha comentat l'existència de sales de multi conferència i s'ha pogut veure el mecanisme de subscripció a aquestes sales. Ara es justificarà la seva existència en contraposició a l'esquema bàsic de senyalització de trucades (INVITE - BYE).

Suposant un escenari amb 4 usuaris que volen iniciar una multi conferència. En primer lloc es pot veure la seqüència de missatges necessaris per iniciar la multi conferència amb missatges INVITE.[1]

[1] En la Fig. 3.5 i en la Fig. 3.6 s'han omès els missatges SIP response i ACK per simplificar l'esquema.

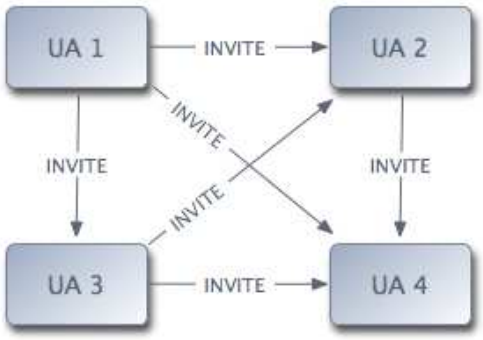


Fig. 3.5 INVITE's entre quatre User Agents SIP

Cada client SIP s'ha d'ocupar d'establir i mantenir la comunicació amb tots els altres. Tots els clients SIP s'han de posar d'acord per enviar-se els INVITE's entre ells sense repeticions (si l'user agent A rep un INVITE de l'user agent B, l'user agent A no ha d'enviar cap altre INVITE a l'user agent B). També s'han d'assegurar que cap d'ells quedi aïllat o no hagi establert la trucada amb tots els altres.

En segon lloc es pot veure la seqüència de missatges necessaris per iniciar la multi conferència amb missatges SUBSCRIBE dirigits al Presence Agent.

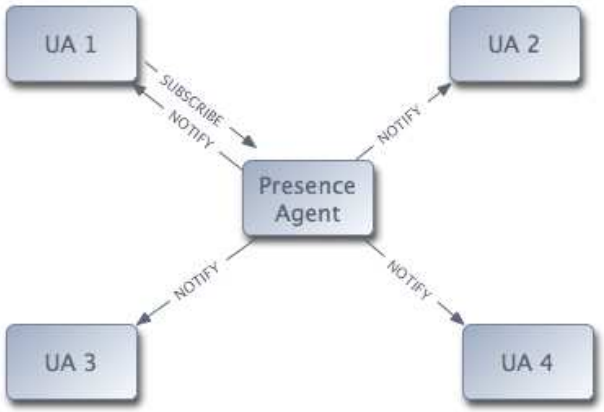


Fig. 3.6 Subscripció i notificació mitjançant el Presence Agent

Cada client SIP només s'ha d'ocupar d'establir i mantenir la comunicació amb el Presence Agent. Si entra un nou usuari a la multi conferència, enviarà un SUBSCRIBE al Presence Agent i ell s'encarregarà de notificar als altres clients.

Amb l'esquema bàsic, si un usuari que vol abandonar la multi conferència ha d'enviar un BYE a cadascun dels membres de la multi conferència, com més usuaris existeixin més retard observarà l'usuari abans de poder abandonar.

Amb l'esquema elegit, l'usuari que vol abandonar la multi conferència només ha d'enviar un SUBSCRIBE al Presence Agent amb la capçalera "Expires" igual a zero. El Presence Agent s'ocupa de notificar a la resta de membres de la multi conferència que l'usuari l'ha abandonat.

Després d'analitzar les possibilitats de cada mètode es va decidir utilitzar les sales de multi conferència. La senzillesa de programació va ser un dels principals motius d'aquesta decisió. També va influir la major escalabilitat i la menor càrrega de cada UA SIP en comparació amb l'altre mètode.

3.2.3. Proxy SIP

Per permetre la comunicació entre B2BUA i inclús entre B2BUA i UA estàndards, s'ha utilitzat l'openSER [5] , un servidor SIP de codi lliure. L'openSER està orientat a la escalabilitat, pot processar centenars de trucades per segon. Està disponible per sistemes Unix/Linux i està escrit en C.

L'openSer pot funcionar com a proxy registrar, servidor de localització i servidor de redireccionament. Incorpora un fitxer de configuració on se l'hi poden afegir funcionalitats i desactivar-ne d'altres. L'openSer utilitzat per aquest treball treballa en base al fitxer de configuració per defecte. Una de les funcionalitats que no implementa és, per exemple, l'autenticació.

Els B2BUA's de l'aplicació i el Presence Agent si que implementen el missatge REGISTER amb autenticació. Han estat testejats amb un altre proxy SIP per comprovar que l'autenticació funciona correctament.

3.2.4. Agent de presència (Presence Agent)

El Presence Agent implementa un User Agent SIP gràcies a la llibreria JAIN SIP, com el B2BUA. Aquest User Agent SIP no té la mateixa funció que els B2BUA's dels usuaris comuns. Aquest no implementa els mateixos missatges SIP i, a més, ha de ser sempre accessible per poder atendre als clients SIP en qualsevol moment.

L'arquitectura interna del Presence Agent és la mateixa que pel B2BUA, les classes principals són SipActor i MySip però en aquest cas s'afegeix una classe Notifier que s'utilitza per la gestió de les subscripcions dels UA's.

Quan rep un missatge SUBSCRIBE, el Presence Agent comprova les capçaleres i crea una classe Notifier per atendre la subscripció de l'usuari. Aquesta classe s'encarrega de comprovar el temps de subscripció i cancel·lar-la si no es rep un altre SUBSCRIBE abans no expiri.

Es considera que l'usuari està *online* quan envia un SUBSCRIBE i *offline* quan envia un SUBSCRIBE amb la capçalera "Expires" de valor zero. Després de

rebre un SUBSCRIBE, canvia l'estat de l'usuari en qüestió i notifica a tots els altres. Així els usuaris tenen sempre una llista de contactes actualitzada, tant de presència en l'aplicació com de presència en multi conferències.

El NOTIFY que s'envia a cada usuari té capçaleres que indiquen l'estat de la subscripció i també conté un XML amb l'estat dels usuaris. Aquest XML té el següent format:

```
<?xml version='1.0' encoding='iso-8859-1'?>

<user_list>
     <user>
          <name>Nom de l'usuari</name>
          <status>online/offline</status>
     </user>
     ...
</user_list>
```

La resta de missatges SIP que implementa el B2BUA com, per exemple, l'INVITE o el BYE no són necessaris pel Presence Agent. La seva funció es limita a la presència, no estableix cap trucada amb un altre User Agent SIP.

3.2.5. Gestor de videoconferències

El WiCat Server conté un mòdul dedicat al control de videoconferències programades pels usuaris. En el capítol anterior es citaven les funcionalitats que permet aquest mòdul, en aquest capítol es mostra la implementació d'aquestes funcionalitats.

El gestor de videoconferències consta de dos classes. La classe *RsrvSession* s'utilitza per guardar la informació d'una videoconferència programada. Conté variables per emmagatzemar la data de la videoconferència, els usuaris que hi participen, el creador de la videoconferència i la descripció d'aquesta.

L'altra classe que pertany a aquest mòdul s'anomena *SessionControl* i té varies funcions:

- S'encarrega d'avisar a l'usuari corresponent quan és l'hora d'iniciar una videoconferència.
- Manté actualitzada la llista de videoconferències programades, conté mètodes per afegir-ne de noves i per esborrar-ne d'antigues.
- Retorna les videoconferències a les quals un usuari en concret està convidat.
- Manté la informació de les videoconferències que hi ha en curs per poder-hi subscriure nous usuaris.
- Conté una classe niuada que utilitza per ordenar per data les videoconferències consultades pels clients.

3.3. Gestió de contingut multimèdia

En el capítol anterior s'ha parlat de la comunicació amb el Flash Media Server i del format propietari Flash Vídeo (FLV). En aquest capítol s'explicarà la manera com s'han posat en pràctica les tècniques de transmissió de contingut multimèdia que proporciona la plataforma del Flash Media Server.

3.3.1. Aplicació del Flash Media Server

Per tal que el Flash Media Server accepti les connexions dels clients Flash, s'ha desenvolupat una aplicació molt senzilla. L'aplicació consisteix bàsicament en un directori creat en el directori *applications* del Flash Media Server. Aquest directori pot contenir fitxers ASC (*server-side* ActionScript) per programar la lògica de l'aplicació. També pot contenir arxius FLV disponibles pels clients, per exemple un vídeo que es vulgui obtenir per *streaming*.

Per desenvolupar una aplicació, els fitxers del client i del servidor s'han de publicar per poder ser accessibles des d'Internet. Per aplicacions web, com és el cas, els fitxers del client, l'aplicació de Flash (SWF) i els HTML necessaris per carregar-la, han d'estar publicats en un directori d'un servidor web.

Els fitxers del servidor, incloent fitxers ASC, fluxos multimèdia (fitxers FLV) i altres fitxers, es publiquen en el directori de l'aplicació registrada en el Flash Media Server.

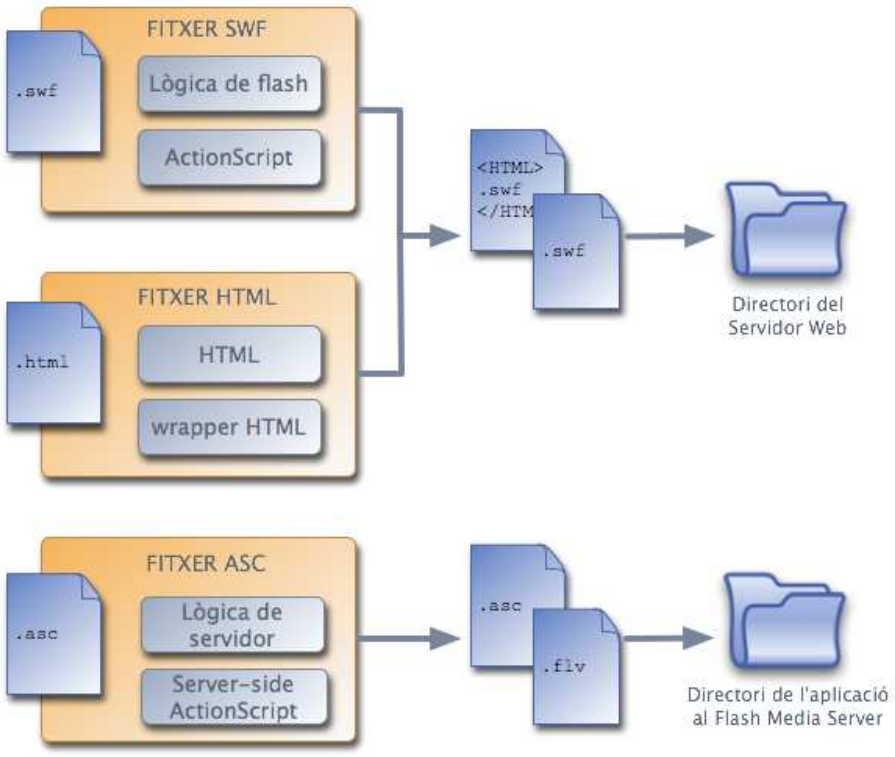


Fig. 3.7 Emplaçament d'arxius per l'aplicació del Flash Media Server

Per aquest TFC s'ha desenvolupat una aplicació senzilla en el FMS[1]. Aquesta aplicació està formada pel directori que li dona nom i, en l'interior, un fitxer main.asc (*server-side* ActionScript) amb un *script* que permet acceptar connexions de clients.

Per l'aplicació WiCat els arxius FLV es generen dinàmicament quan els clients els publiquen, i s'allotgen en el directori de l'aplicació registrada en el FMS. Quan els clients es desconnecten del Flash Media Server s'eliminen.

Quan un usuari executa el fitxer SWF en el seu navegador i el SWF es connecta al servidor, el FMS carrega l'aplicació i, si no n'hi ha cap executant-se, en crea una instància (crida *onAppStart*). El servidor crea un objecte Client per representar l'aplicació del client en el servidor i accepta la connexió. En el client s'invoca *onStatus* per informar si la connexió ha estat acceptada o no. Quan el client tanca la connexió, es crida *onDisconnect* en el servidor. Quan l'aplicació es para, s'invoca *onAppStop*. Aquest flux s'observa en el diagrama següent (Fig. 3.8).

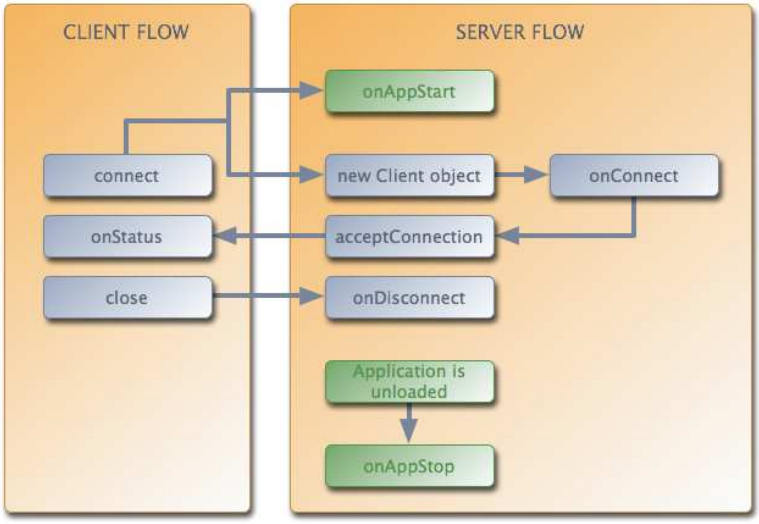


Fig. 3.8 Flux de comunicació client - servidor

3.3.2. Aplicació del client Flash

En el client de Flash és on resideix la "intel·ligència" de l'aplicació pel que fa a la transmissió multimèdia. El client es connecta al Flash Media Server i, a partir d'aquí, s'encarrega de subscriure's als fluxos corresponents i de publicar el seu flux. El FMS només accepta la seva connexió i s'encarrega d'allotjar els fluxos de vídeo en temps real perquè els usuaris els puguin reproduir.

Per la comunicació amb el Flash Media Server, reproducció dels vídeos i captura de la webcam i del micròfon, s'ha utilitzat un arxiu SWF dedicat. Aquest

[1] Flash Media Server (Consultar els ACRÒNIMS)

arxiu s'anomena call.swf i es carrega en la part central de l'escenari cada cop que s'inicia una videoconferència.

L'arxiu call.swf proporciona una funcionalitat afegida a l'aplicació, permet canviar el nombre d'usuaris d'una videoconferència dinàmicament. Això vol dir que si un usuari abandona la videoconferència, automàticament s'eliminarà el vídeo d'aquest usuari de l'escenari i s'ampliarà la resolució de la resta de vídeos. De la mateixa manera, si un usuari s'afegeix a la videoconferència es reduirà la resolució de tots els vídeos i apareixerà el nou vídeo en l'escenari.

Per implementar aquesta funcionalitat l'arxiu call.swf conté una funció que s'encarrega d'establir la connexió amb el Flash Media Server, publicar el flux de l'usuari local i subscriure's als fluxos de la resta d'usuaris. Aquesta funció es crida cada cop que es rep una notificació del Presence Agent, així el client es re connecta al Flash Media Server i es subscriu als usuaris que correspongui en aquell moment.

3.5. Escenari de proves

3.5.1. Instal·lació d'equips

En el diagrama següent s'observa la distribució dels equips necessaris pel funcionament complet de la plataforma de videoconferències.

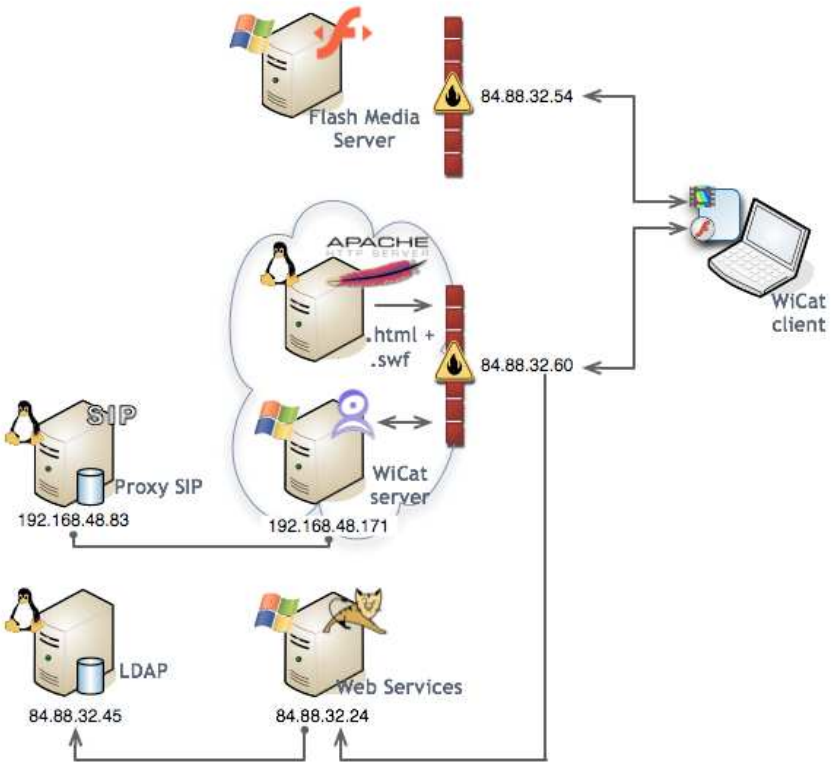


Fig. 3.9 Emplaçament d'equips

El primer equip que s'observa és el servidor que conté el Flash Media Server 2, amb la IP pública 84.88.32.54. Aquest equip està protegit per un tallafocs, que té obert el port 1935. Per defecte, el FMS escolta en el port 1935, ja que és l'assignat per la IANA[1]. És possible configurar el servidor perquè escolti en un altre port, per exemple el port 80 o el 443. Si s'utilitza aquesta opció s'ha de tenir en compte que si hi ha un servidor web en la mateixa màquina, escoltant en el mateix port, hi haurà un conflicte.[2]

Un dels equips més important és el que conté el servidor web Apache i el WiCat Server. El servidor web serveix per obtenir els fitxers HTML i SWF i executar l'aplicació Flash en el navegador del client. Un cop executant-se en el navegador, el fitxer SWF principal realitza una connexió TCP amb el WiCat Server i s'hi comunica mitjançant el protocol propi XML.

El servidor web i l'aplicació es troben en una DMZ[3]. Així per accedir a tots dos servidors s'utilitzarà la mateixa IP pública (84.88.32.60). Una de les restriccions de seguretat que imposa el Flash és que els fitxers SWF i el servidor que rep les connexions d'aquests SWF han de residir en el mateix subdomini. El WiCat Server escolta en el port 9999, el tallafocs que protegeix la DMZ permet connexions en el port 9999 i el 80.

La IP privada 192.168.48.171 de l'equip que conté el WiCat Server s'utilitza per la comunicació de la Pila SIP amb el proxy SIP ubicat en la xarxa privada. Al trobar-se en la mateixa xarxa, s'eviten problemes al travessar NAT's[4] en les comunicacions SIP.

Per la funcionalitat de gestió d'usuaris i equips de videoconferència, l'aplicació fa peticions HTTP-SOAP al servei web. Es pot accedir a aquest servei web en la màquina amb IP pública 84.88.32.24. Aquesta màquina disposa d'un servidor d'aplicacions Apache Tomcat on es desplega el servei web mitjançant AXIS. La raó d'aquesta configuració és permetre l'accés al servei web des d'Internet i facilitar la col·laboració entre diferents serveis web de vàries corporacions.

El directori LDAP resideix en l'equip amb IP pública 84.88.32.45. Amb l'arquitectura utilitzada podria instal·lar-se en una xarxa privada, sempre i quant el servei web hi pugui accedir. Concretament s'ha utilitzat una implementació oberta de LDAP, l'OpenLDAP per indexar la informació dels usuaris i equips de videoconferència.

3.5.2. Proves realitzades amb l'aplicació

L'escenari de proves utilitzat consta de tres usuaris amb els seus corresponents clients Flash i el Flash Media Server com a element d'interconnexió entre ells.

[1] Internet Assigned Numbers Authority (Consultar els ACRÒNIMS)
[2] Per més informació sobre les posibles configuracions del Flash Media Server veure l'annex B.4.
[3] Demilitarized Zone (Consultar els ACRÒNIMS)
[4] Network Address Translation (Consultar els ACRÒNIMS)

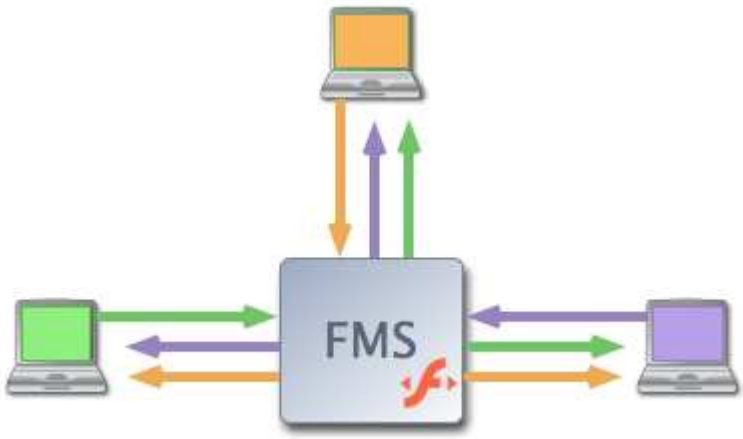


Fig. 3.10 Escenari de videoconferència amb tres usuaris

En l'esquema s'observa que amb tres clients, el Flash Media Server maneja nou fluxos, tres d'entrada i sis de sortida. Com més usuaris participin en la videoconferència, més ample de banda consumirà el Flash Media Server. En la Taula 3.3 s'observa la relació de fluxos d'entrada i sortida del FMS segons el numero d'usuaris que participin en la videoconferència.

Taula 3.3 Relació de fluxos d'entrada i sortida del Flash Media Server

nº usuaris	entrada	sortida	relació	total
2	2 fluxos	2 fluxos	out = in	4 fluxos
3	3 fluxos	6 fluxos	out = 2*in	9 fluxos
4	4 fluxos	12 fluxos	out = 3*in	16 fluxos
5	5 fluxos	20 fluxos	out = 4*in	25 fluxos
n	n fluxos	n*(n-1) fluxos	out = (n-1)*in	n^2 fluxos

S'observa que hi ha una relació entre en nombre d'usuaris i els fluxos d'entrada i sortida del Flash Media Server. En el gràfic següent s'observa la transmissió dels fluxos RTMP dels usuaris cap al FMS. L'eix x mostra el temps de la captura (en total 200 segons), l'eix y mostra l'ample de banda en Bytes per segon. En la llegenda del gràfic s'especifica l'ample de banda promig de cada client.

Fig. 3.11 Ample de banda d'entrada al FMS

L'ample de banda que consumeixen els fluxos RTMP dels tres clients són gairebé iguals. Per això es pot suposar que la relació entre fluxos d'entrada i sortida del FMS també es complirà per l'ample de banda.

BWout = (n-1)*BWin.

Si es calcula la suma dels amples de banda d'entrada al FMS:

139,092 Kbps + 165,847 Kbps + 118,20 Kbps = **423,139 Kbps**

L'ample de banda promig d'entrada al Flash Media Server és de 423,139 Kbps. Perquè la relació es compleixi, l'ample de banda de sortida del FMS hauria de ser el doble que el d'entrada, per comprovar-ho es mostra una captura de l'ample de banda de sortida del FMS:

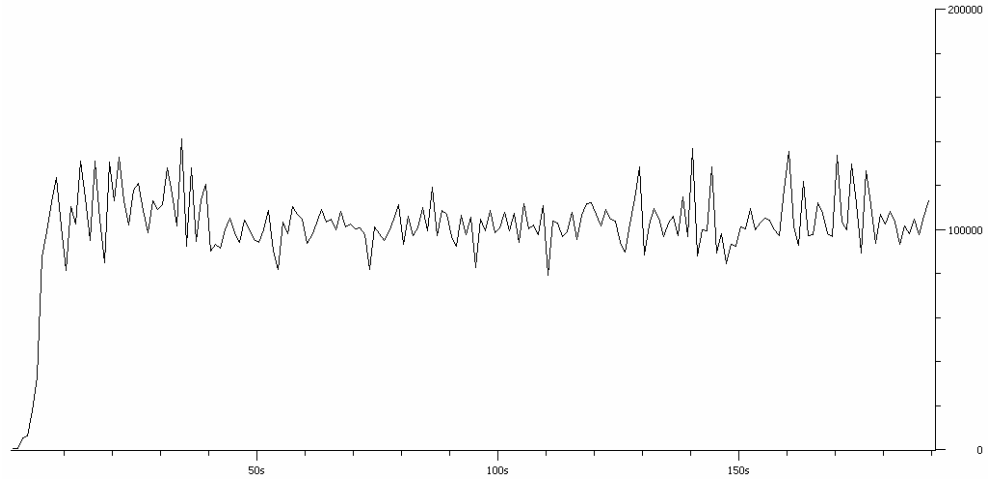


Fig. 3.12 Ample de banda de sortida del FMS

L'ample de banda promig durant tota la captura és de **797,267 Kbps** (com mostra el gràfic, 99658,429 Bps). Si es calcula la relació entre l'ample de banda d'entrada (423,139 Kbps) i el de sortida dóna el següent resultat:

797,267 Kbps / 423,139 Kbps = **1.88**

S'observa que el factor s'aproxima molt a 2, per tant, podem afirmar que la relació es compleix. Això passa perquè l'ample de banda de sortida de cada client és quasi el mateix, ja que els paràmetres de la captura són iguals, els que Flash proporciona per defecte:

- La resolució està fixada a 120 x 160 píxels.
- La velocitat de captura a 15 fps.
- L'ample de banda de codificació màxim a 131,072 Kbps.
- La qualitat, és a dir, el factor de compressió és variable. Augmenta o disminueix per no superar l'ample de banda màxim segons els paràmetres anteriors.

Les diferències entre els amples de banda promig d'enviament de cada client tenen a veure amb el tipus de captura realitzada. Depenent de la quantitat de moviment que captura la webcam i de la rapidesa d'aquest moviment, l'ample de banda promig varia i s'observen més o menys pics en la gràfica.

De l'estudi també es pot extreure l'ample de banda total necessari pel Flash Media Server i també el necessari per cada client segons el nombre d'usuaris i l'ample de banda de codificació:

$$\mathbf{BWs = N^2 * S} \qquad \mathbf{BWc = N * S}$$

On N correspon al nombre d'usuaris de la videoconferència i S és el flux codificat a velocitat constant. En realitat per les oscil·lacions que s'observen en la gràfica es pot suposar que la velocitat de codificació no és constant sinó que varia en funció de la quantitat de moviment capturada. El *codec* que utilitza el Flash Player per la codificació de les dades de vídeo, On2 VP6 [21] , és propietari i les especificacions tècniques no estan disponibles. Degut a aquesta limitació no és possible calcular el paràmetre S. A més per calcular aquest paràmetre també s'hauria de tenir en compte el format dels paquets RTMP, dels quals tampoc es disposa d'informació.

CAPÍTOL 4. PLANIFICACIÓ

Aquest capítol mostra les diferents tasques realitzades per aconseguir l'objectiu final d'aquest treball i la divisió d'aquestes tasques en el temps de dedicació total del treball. Segons el tipus de tasques s'han dividit en cinc grups:

Estudi previ: Aquesta tasca engloba la feina de recerca d'informació, estudi de protocols i tecnologies a utilitzar i desenvolupament de prototips senzills per provar les opcions disponibles.

Disseny: En aquest grup entren les tasques de disseny de protocols, de software i definició de l'arquitectura. També el disseny gràfic de l'aplicació web.

Implementació: Engloba totes les tasques relatives a la programació de software seguint el disseny.

Testeig: Aquesta tasca correspon a les proves fetes durant la implementació i les proves finals.

Documentació: La redacció de la memòria i dels documents fets durant la realització del treball entren dins d'aquesta tasca.

4.1. Temps de dedicació

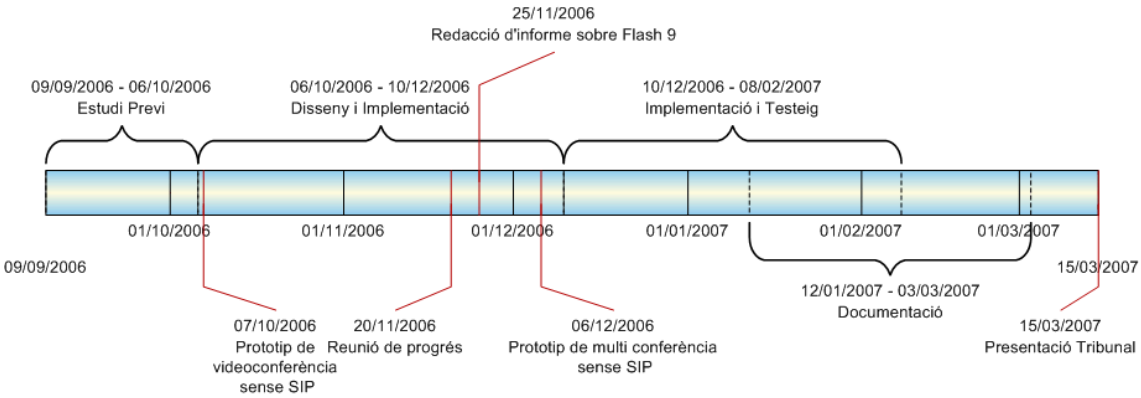


Fig. 4.1 Escala de temps del treball

Com es pot comprovar en l'escala de temps, les tasques s'han anat solapant les unes amb les altres al llarg de tota la durada del treball. El desconeixement inicial de la plataforma de Flash i Flash Media Server, va comportar que el disseny s'anés especificant a mesura que la corba d'aprenentatge de les tecnologies utilitzades anava creixent. La programació de diferents prototips per separat i la consegüent integració va provocar que es realitzessin proves de funcionament durant el període d'implementació.

4.2. Tasques realitzades

La següent taula mostra les tasques realitzades al llarg de tot el treball, classificades per tipus i amb el temps de dedicació de cadascuna.

Taula 4.1 Tasques realitzades

Tipus de tasca	Descripció	Temps invertit
Estudi previ	Establir requeriments i buscar informació.	40 hores
Estudi previ	Implementar un prototip en Flash que capturi d'una web cam i ho mostri.	20 hores
Estudi previ	Instal·lar el Flash Media Server i implementar-hi una aplicació senzilla que accepti connexions de clients Flash.	20 hores
Implementació	Implementar les funcions necessàries per fer consultes en un LDAP mitjançant JNDI.	30 hores
Implementació	Desenvolupar el servei web a partir de les funcions ja implementades i desplegar-lo.	40 hores
Implementació	Ampliar el prototip inicial perquè capturi d'una web cam, ho enviï al FMS i ho mostri amb streaming.	15 hores
Estudi previ	Desenvolupar un servidor en Java i un client Flash que es comuniquin mitjançant XML.	15 hores
Disseny	Dissenyar el protocol de comunicació entre el client Flash i el servidor en Java basat en XML.	3 hores
Implementació	Desenvolupar un client Flash on es pugui triar una data, hora, participants i descripció i comunicar-lo amb el servidor Java.	8 hores
Implementació	Implementar les funcions bàsiques del gestor de videoconferències.	6 hores
Implementació	Desenvolupar un client Flash que consulti el servidor i mostri les videoconferències programades.	4 hores
Implementació	Implementar un UA SIP simple i comunicar-lo amb el servidor Java.	35 hores
Implementació	Realitzar un prototip per fer videoconferències de dos participants amb senyalització.	15 hores
Disseny i implementació	Implementar un prototip de Presence Agent per la presència d'usuaris en l'aplicació.	20 hores
Disseny i implementació	Dissenyar i implementar un client Flash que, mitjançant el Web Service i el Presence Agent, obtingui la llista d'usuaris i mostri el seu estat.	15 hores
Implementació i testeig	Afegir el missatge REGISTER als B2BUA i al Presence Agent i fer proves amb el proxy SIP.	15 hores
Implementació	Afegir autenticació a la web i integrar amb la presència, les trucades i la programació de videoconferències mitjançant el LocalConnection.	35 hores

Disseny	Dissenyar el sistema de multi conferències.	7 hores
Implementació	Afegir funcionalitats al gestor de videoconferències i programar la resta de mòduls que intervenen en les multi conferències programades.	40 hores
Implementació	Implementació de les multi conferències no programades.	40 hores
Disseny	Retocar el disseny gràfic de l'aplicació, triar un nom i confeccionar un logotip.	30 hores
Testeig	Fer proves de totes les funcionalitats i mesures de rendiment, ample de banda,etc.	15 hores
Documentació	Redacció de la memòria del treball.	120 hores

El total d'hores dedicades al desenvolupament del treball han estat 588. Aquest numero és més alt que el recomanat per la realització d'un TFC (sobre les 500). La raó d'aquesta alta dedicació és l'ús de varies tecnologies diferents. Cadascuna requereix un estudi, implementació i posterior integració amb la resta del sistema.

En la taula es pot observar que el període d'establir requeriments i d'estudi previ ha resultat més llarg que moltes altres tasques del treball. Això es deu a la difícil compaginació que comporta tenir una aplicació web capaç de rebre i enviar *streaming* de vídeo i disposar, a més, de senyalització per les trucades.

La corba d'aprenentatge de l'entorn de desenvolupament d'Adobe Flash 8 i del llenguatge ActionScript 2, ha suposat un endarreriment important a l'hora de desenvolupar petites parts de la web que haurien d'haver suposat molt poc temps.

La tasca que ha portat més entrebancs i endarreriments ha estat el desplegament del servei web. La programació va ser relativament ràpida tot i l'aprenentatge d'una llibreria desconeguda fins llavors, la API JNDI. El que va portar més complicacions va ser preparar l'entorn de l'Apache AXIS i desplegar el servei web perquè el client hi pogués fer peticions.

La utilització de la API JAIN SIP també va suposar un temps d'aprenentatge important, sobretot la part del Presence Agent ja que existeix poca informació sobre com implementar un mòdul d'aquest tipus. Tota la informació va ser extreta del RFC 3265 (SIP-Specific Event Notification).

CAPÍTOL 5. CONCLUSIONS

5.1. Objectius assolits

Al inici d'aquest TFC es van definir les funcionalitats desitjades per l'aplicació. Assolir aquestes funcionalitats suposava un repte, ja que requeria l'estudi de varies tecnologies. El poc coneixement d'aquestes tecnologies era un inconvenient a l'hora d'avaluar les possibilitats de cadascuna. Per això es va fer una recerca i posterior selecció de les més adequades per complir els objectius inicials.

L'objectiu principal que es volia assolir, com il·lustra el títol del TFC, és la realització de videoconferències en un entorn web. A més també es requeria que les videoconferències es recolzessin en un protocol de senyalització com el SIP.

Per dur a terme l'objectiu principal es va triar l'ús de les tecnologies oferides per Adobe Flash. Aquestes, tot i ser propietàries i de pagament, han simplificat i agilitzat la realització de videoconferències en un entorn web. També han sigut d'utilitat a l'hora de desenvolupar la interfície web, perquè fos simple i amigable per l'usuari.

L'ús de la API JAIN SIP i la utilització d'un protocol propi per la comunicació entre client i servidor, han permès la integració del protocol SIP al sistema de videoconferències. A més ha ajudat a complir altres objectius opcionals, com la presència d'usuaris en l'aplicació i la realització de multi conferències.

La gestió de videoconferències i, d'usuaris i equips de videoconferència han servit perquè l'usuari pugui obtenir informació sobre el personal de la seva empresa i per poder programar videoconferències amb ells. Aquestes dues funcionalitats s'han dut a terme mitjançant un mòdul del servidor desenvolupat en Java i l'ús de serveis web.

5.2. Impacte mediambiental

Tot i que a primer cop d'ull no sembla important l'impacte mediambiental d'aquest treball, a continuació es demostra el contrari.

Les videoconferències permeten la comunicació entre usuaris allunyats geogràficament. Sobretot en l'àmbit empresarial, moltes vegades és necessari reunir empleats de diferents sucursals, sovint aquests empleats són de països diferents.

Reunir persones de llocs diferents suposa un impacte mediambiental considerable, si no es fa us de la videoconferència. Cada empleat hauria de

viatjar per reunir-se amb els demés. Si es viatja en cotxe suposa un consum de gasolina o gasoil, per tant es contribueix a l'esgotament d'una font d'energia no renovable, el petroli. A més d'això, cada litre de gasoil allibera 2,67 Kg de diòxid de carboni, CO2, i un litre de gasolina n'allibera 2,13[1] teòricament (en la pràctica la combustió no és perfecta). Aquest gas contaminant és un dels principals causants de l'escalfament global del planeta.

Si els usuaris són de països diferents probablement viatgin en avió. A part del consum de combustible i emissió de gasos contaminants, un avió genera una estela de vapor d'aigua a molta altura. Algunes d'aquestes esteles persisteixen durant hores i es comporten de la mateixa manera que els núvols cirrus de gran altura, atrapant el calor de l'atmosfera i contribuint a l'escalfament global.

Utilitzant l'aplicació de videoconferències, es pot reunir personal allunyat geogràficament, sense necessitat d'utilitzar mitjans de transport que contaminen el medi ambient i contribueixen en el canvi climàtic.

5.3. Conclusions personals

La realització d'aquest treball, m'ha aportat coneixements i experiència en l'elaboració de projectes. Al inici tenia uns coneixements limitats de SIP, no havia utilitzat mai l'entorn de Flash i no sabia com funcionaven els serveis web. Al final d'aquest TFC puc dir que sóc capaç d'utilitzar totes tres tecnologies en profunditat.

Gràcies al llenguatge de programació orientat a objectes Java, s'han pogut integrar les diferents tecnologies amb facilitat i, sobretot, rapidesa. Aquest és un llenguatge que ja coneixia i havia utilitzat, la realització del TFC m'ha ajudat a ampliar coneixements i a aprendre l'ús de varies llibreries.

Ha estat molt interessant utilitzar la plataforma d'Adobe Flash per la transmissió multimèdia. És una tecnologia emergent que aporta molts avantatges per la realització de videoconferències i que ofereix moltes altres possibilitats que m'agradaria continuar estudiant.

5.4. Treballs futurs

Els requeriments inicials pel sistema de videoconferències han estat complerts. Tot i això existeixen millores interessants per afegir i que resultarien de gran utilitat.

- Implementació d'un proxy RTP-RTMP.

[1] Dades extretes d'un article de "Ecologistas en acción". URL: http://www.ecologistasenaccion.org/article.php3?id_article=5580

- Poder triar des del client l'ample de banda desitjat per transmetre els fluxos multimèdia.

- Afegir la opció de ser usuari actiu o passiu (transmetre vídeo i àudio o només visualitzar els dels demés).

- Afegir estats als contactes apart de connectat i no connectat (absent, no disponible,etc).

- Afegir un xat de text pels usuaris de la videoconferència.

- Implementar un mesurador de les capacitats de la connexió de l'usuari, i fixar els paràmetres de captura segons aquest valor.

BIBLIOGRAFIA

[1] **VoIP:** Notícies sobre software i hardware per VoIP i vídeo sobre IP. (en línia). [Última Consulta: 12 de novembre de 2006]. URL: http://www.voipnow.org/

[2] **ViDe:** Informació sobre videoconferència. Usos, tecnologies, tècniques, exemples, etc. (En línia). [Última Consulta: 15 de novembre de 2006]. URL:http://www.vide.net/cookbook/

[3] **SIP:** RFC 3261. Session Initiation Protocol, documentació de referència. (En línia). [Última Consulta: 20 de novembre de 2006]. URL: http://www.ietf.org/rfc/rfc3261.txt

[4] **SIP Specific Event Notification:** RFC 3265. Extensió de SIP que defineix la manera com els nodes SIP demanen notificacions a nodes remots per saber l'estat de certs events. (En línia). [Última Consulta: 25 de novembre de 2006]. URL: http://www.ietf.org/rfc/rfc3265.txt

[5] **OpenSER:** SIP server de codi obert. (En línia)[Última Consulta: 27 de novembre de 2006]. URL: http://www.openser.org/

[6] **JAIN SIP:** Web oficial de JAIN SIP, informació de la llibreria, codi font, descarrega de fitxers binaris. (En línia). [Última Consulta: 12 de novembre de 2006]. URL: https://jain-sip.dev.java.net/

[7] **H323:** Web amb informació sobre el protocol H323. RFC's, novetats, fòrums, documents d'interès, etc. (En línia). [Última Consulta: 16 de setembre de 2006]. URL: http://www.packetizer.com/voip/h323/

[8] **JAVA Sun:** Web oficial de Java Sun. (En línia). [Última Consulta: 2 de febrer de 2007]. URL: http://java.sun.com/

[9] **JMF:** Web oficial del projecte Java Media Framework. API, fòrums, descàrregues i exemples. (En línia). [Última Consulta: 30 de setembre de 2006]. URL: http://java.sun.com/products/java-media/jmf/

[10] **Macromedia Flash Support Center:** Informació útil per desenvolupadors de Flash, API's, tutorials, fòrums,etc. (En línia). [Última Consulta: 4 de febrer de 2007]. URL: http://www.adobe.com/support/flash/

[11] **Cristalab:** Tutorials de Flash en castellà, tant per desenvolupadors com per dissenyadors. (En línia). [Última Consulta: 20 de desembre de 2006]. URL: http://www.cristalab.com

[12] **Flash Media Server:** Web oficial del servidor de Flash. Descàrrega de fitxers binaris, manual d'instal·lació, notícies, especificació,etc. (En línia). [Última Consulta: 15 de febrer de 2007]. URL: http://www.adobe.com/products/flashmediaserver/

[13] **Flash Media Server Install:** Guia d'instal·lació del Flash Media Server, per Windows i Linux. Obtingut amb la instal·lació del servidor. [Última Consulta: 15 de febrer de 2007].

[14] **Flash Media Server Developing:** Guia de desenvolupament d'aplicacions utilitzant la plataforma del Flash Media Server. Obtingut amb la instal·lació del servidor. [Última Consulta: 15 de febrer de 2007].

[15] **Flash Media Server Managing:** Guia d'administració del Flash Media Server. Obtingut amb la instal·lació del servidor. [Última Consulta: 15 de febrer de 2007].

[16] **FMS client-side ActionScript Language Reference:** Classes ActionScript de client per la comunicació amb el Flash Media Server. Explicació i exemples. (En línia). [Última Consulta: 2 de febrer de 2007]. URL: http://livedocs.adobe.com/fms/2/docs/wwhelp/wwhimpl/js/html/wwhelp.htm?href=Part_SS_ASD.html

[17] **Flash Video Learning Guide:** Article del Flash Developer Center on s'orienta al desenvolupador en les diferents opcions i bones pràctiques per utilitzar el Flash Video. (En línia). [Última Consulta: 14 de desembre de 2006]. URL: http://www.adobe.com/devnet/flash/articles/video_guide.html

[18] **Encoding Best Practices for live Video:** Article del Flash Developer Center on es fan recomanacions per optimitzar la distribució de vídeo en viu tenint en compte els requeriments d'ample de banda. (En línia). [Última Consulta: 17 de febrer de 2007]. URL: http://www.adobe.com/devnet/flash/articles/flv_live.html

[19] **Attacks and Firewalls in FMS:** Article del Flash Communication Server Developer Center. Ofereix una guia per protegir el Flash Media Server d'atacs i per accedir a ell des de l'interior d'un tallafocs. (En línia). [Última Consulta: 15 de febrer de 2007]. URL: http://www.adobe.com/devnet/flashcom/articles/firewalls_proxy02.html

[20] **ActionScript 3:** Introducció a la nova llibreria de Flash 9, instal·lació, tutorials, API's, exemples, etc. (En línia). [Última Consulta: 12 de desembre de 2006]. URL: http://labs.adobe.com/wiki/index.php/ActionScript_3

[21] **On2 VP6:** Pàgina oficial de On2 Technologies. Descripció de les funcionalitats del codec utilitzat per Adobe Flash. (En línia). [Última Consulta: 1 de març de 2006]. URL: http://www.on2.com/technology/vp6/

[22] **Red5:** Informació sobre el servidor de Flash de codi lliure. Descàrrega de binaris i codi font, tutorials, llibreries, notícies,etc. (En línia). [Última Consulta: 7 de desembre de 2006]. URL: http://osflash.org/red5/

[23] **OpenLdap:** Web amb un manual d'instal·lació i configuració del servei de directori OpenLdap. (En línia). [Última Consulta: 1 de novembre de 2006]. URL: http://fferrer.dsic.upv.es/cursos/Linux/Avanzado/HTML/ch02s04.html

[24] **JNDI:** Tutorial de la API de Java JNDI amb exemples de codi. (En línia). [Última Consulta: 10 de novembre de 2006]. URL: http://java.sun.com/products/jndi/tutorial/trailmap.html

[25] **Log4j:** Introducció a la API de *logging* d'Apache. Tutorial, exemples, etc. (En línia). [Última Consulta: 1 de novembre de 2006]. URL: http://logging.apache.org/log4j/docs/manual.html

[26] **Web Services-Axis:** Web del projecte d'Apache per desenvolupar serveis web. Manual d'instal·lació, guia de desenvolupament, descàrrega de la API, etc. (En línia). [Última Consulta: 5 de febrer de 2007]. URL: http://ws.apache.org/axis/

[27] **Directory Services Middelware for Multimedia Conferencing:** Article on s'explica l'ús de l'estàndard H350 que descriu l'arquitectura de directori per conferències multimèdia fent servir LDAP. (En línia). [Última Consulta: 5 de febrer de 2007]. URL: http://lab.ac.uab.edu/vnet/cookbook/v2.1/index.html

Alguns dels conceptes definits, s'han obtingut de: *Wikipedia the free encyclopedia that anyone can edit*. (En línia) [Última Consulta: 27 de febrer de 2007]. URL: http://en.wikipedia.org/wiki/Main_Page

ACRÒNIMS

API

Application **P**rogramming **I**nterface. És un conjunt d'especificacions per la comunicació entre components software.

B2BUA

Back to **B**ack **U**ser **A**gent. Actua com un User Agent pels dos extrems d'una trucada SIP. El B2BUA es responsable de manejar la senyalització SIP entre els extrems d'una trucada, des de l'establiment fins l'acabament.

CRC

Class **R**esponsibility **C**ollaborator**.** Les targetes CRC són una metodologia pel disseny de software orientat a objectes.

DMZ

De**m**ilitarized **Z**one. En una area de xarxa (una subxarxa) la DMZ està entre la xarxa interna de la organització i una xarxa externa, normalment Internet.

FLV

Flash **V**ideo. És un format de fitxer propietari per transmetre vídeo a través d'Internet fent servir l'Adobe Flash Media Server i l'Adobe Flash Player.

FMS

Flash **M**edia **S**erver (anteriorment anomenat Flash Communication Server) és un servidor de dades i multimèdia d'Adobe Systems Inc. (originalment un producte de Macromedia). Aquest servidor actua junt amb el Flash Player per crear comunicació multimèdia i RIA's (Rich Internet Applications) multi usuari.

HTML

Hyper**T**ext **M**arkup **L**anguage. És el llenguatge de marcat d'hipertext predominant per la creació de pàgines Web.

HTTP

Hyper**T**ext **T**ansfer **P**rotocol. Protocol que s'utilitza per transmetre informació en el World Wide Web. Principalment es fa servir per obtenir pagines HTML.

IANA

Internet **A**ssigned **N**umbers Authority. És l'entitat que supervisa l'assignació global d'adreces IP, l'administració de zones arrel de DNS i altres assignacions de protocols d'Internet.

IETF

Internet **E**ngineering **T**ask **F**orce. Organització que desenvolupa i promou estàndards d'Internet. És una organització oberta, de voluntaris, sense membres formals ni requeriments per formar-ne part.

IP

Internet **P**rotocol. És un protocol de la capa de xarxa que s'utilitza per comunicar dades a través d'una xarxa de paquets. IP aporta un servei d'adreces úniques i globals per la comunicació entre equips.

ITU-T

ITU Telecommunication Standardization Sector. Coordina els estàndards per telecomunicacions en nom de la International Telecommunication Union (ITU). Els estàndards que produeix la ITU-T són "recomanacions" que suposen un major reconeixement formal internacional que la majoria d'organitzacions que publiquen especificacions tècniques de manera similar.

JMF

Java **M**edia **F**ramework. És una llibreria de Java que permet afegir a les aplicacions Java i als applets àudio i vídeo. Pot capturar, reproduir, transmetre i codificat múltiples formats multimèdia. Estén la Java SE (Java Platform, Standard Edition).

JNDI

Java **N**aming and **D**irectory **I**nterface. És una API per serveis de directori que permet als clients consultar i modificar objectes de dades mitjançant el nom.

LDAP

Lightweight **D**irectory **A**ccess **P**rotocol. És un protocol per consultar i modificar serveis de directori accessibles per TCP/IP. Un directori és un conjunt d'informació amb atributs similars, organitzat d'una manera lògica i jeràrquica.

NAT

Network **A**ddress **T**ranslation. S'encarrega de reescriure l'adreça origen i/o destí dels paquets IP quan passen a través d'un router o un tallafocs. Normalment s'utilitza per donar accés a múltiples equips d'una xarxa privada fent servir una única IP pública.

PA

Presence **A**gent. Element SIP desenvolupat en aquest TFC encarregat de gestionar la subscripció i notificació d'events dels User Agents SIP.

RFC

Request **F**or **C**omments. Són documents amb una sèrie de memoràndums que inclouen noves investigacions, innovacions i metodologies aplicables a tecnologies d'Internet.

RTMP

Real **T**ime **M**essaging **P**rotocol. És un protocol propietari desenvolupat per Adobe Systems (primerament desenvolupat per Macromedia) que s'utilitza principalment amb el Flash Media Server per transmetre àudio i vídeo a través d'Internet cap al client Adobe Flash Player.

RTP

Real-time **T**ransport **P**rotocol. Protocol de la capa d'aplicació que defineix un format de paquet estàndard per transmetre àudio i vídeo a través d'Internet.

SIP

Session **I**nitiation **P**rotocol. És un protocol de la capa d'aplicació (senyalització) per crear, modificar i acabar sessions amb un o més participants. Aquestes sessions inclouen trucades per Internet, distribució multimèdia i conferències multimèdia.

SOAP

Simple **O**bject **A**ccess Protocol. És un protocol de la capa d'aplicació per intercanviar missatges basats en XML entre equips de xarxa, normalment fent servir HTTP.

SWF

Small **W**eb **F**ormat. És el format de fitxer propietari d'Adobe Flash. Pensat per ser prou petits per la publicació Web, els fitxers SWF poden contenir animacions o applets de varis graus d'interactivitat i funcionalitat. Són compilats i comprimits a partir dels fitxers editables .fla (amb els que es treballa en Flash).

TCP

Transmission **C**ontrol **P**rotocol. És un protocol de la capa de transport que s'utilitza per establir connexions entre equips de la mateixa xarxa. Aquests equips es poden transmetre fluxos de dades mitjançant sockets.

UA

User **A**gent. En VoIP, una aplicació client en un sistema SIP s'anomena SIP user agent. Un UA és la combinació d'un UAC i un UAS. El SIP user agent permet trucades peer-to-peer fent servir un protocol client-servidor.

UAC

User **A**gent **C**lient. En SIP l'user agent client inicia una SIP request que envia cap al UAS.

UAS

User **A**gent **S**erver. En SIP l'user agent server accepta les SIP *requests* d'un UAC i genera una resposta *accept*, *reject* o *redirect* de l'usuari.

VoIP

Voice **o**ver **IP**. És l'encaminament de converses de veu sobre Internet o sobre qualsevol altra xarxa basada en IP.

WS

Web **S**ervices. És un software designat per aportar una interacció entre equips d'una xarxa. Normalment són API's que poden ser accedides a través d'Internet.

WSDL

Web **S**ervices **D**escription **L**anguage. És un llenguatge basat en XML que proporciona un model per descriure serveis Web.

XML

e**X**tensible **M**arkup **L**anguage. Llenguatge de marcat ampliable o extensible. Desenvolupat pel World Wide Web Consortium (W3C).

ANNEXES

ANNEX A. WICAT CLIENT

Aquest annex explica amb detall la lògica utilitzada per la implementació del WiCat Client. Es mostren captures realitzades durant l'ús de l'aplicació i s'expliquen les classes d'ActionScript 2 utilitzades per la implementació del client web.

A.1. Flux de l'aplicació

A continuació es mostra un diagrama de flux on es pot observar amb detall el disseny de l'aplicació Flash WiCat Client.

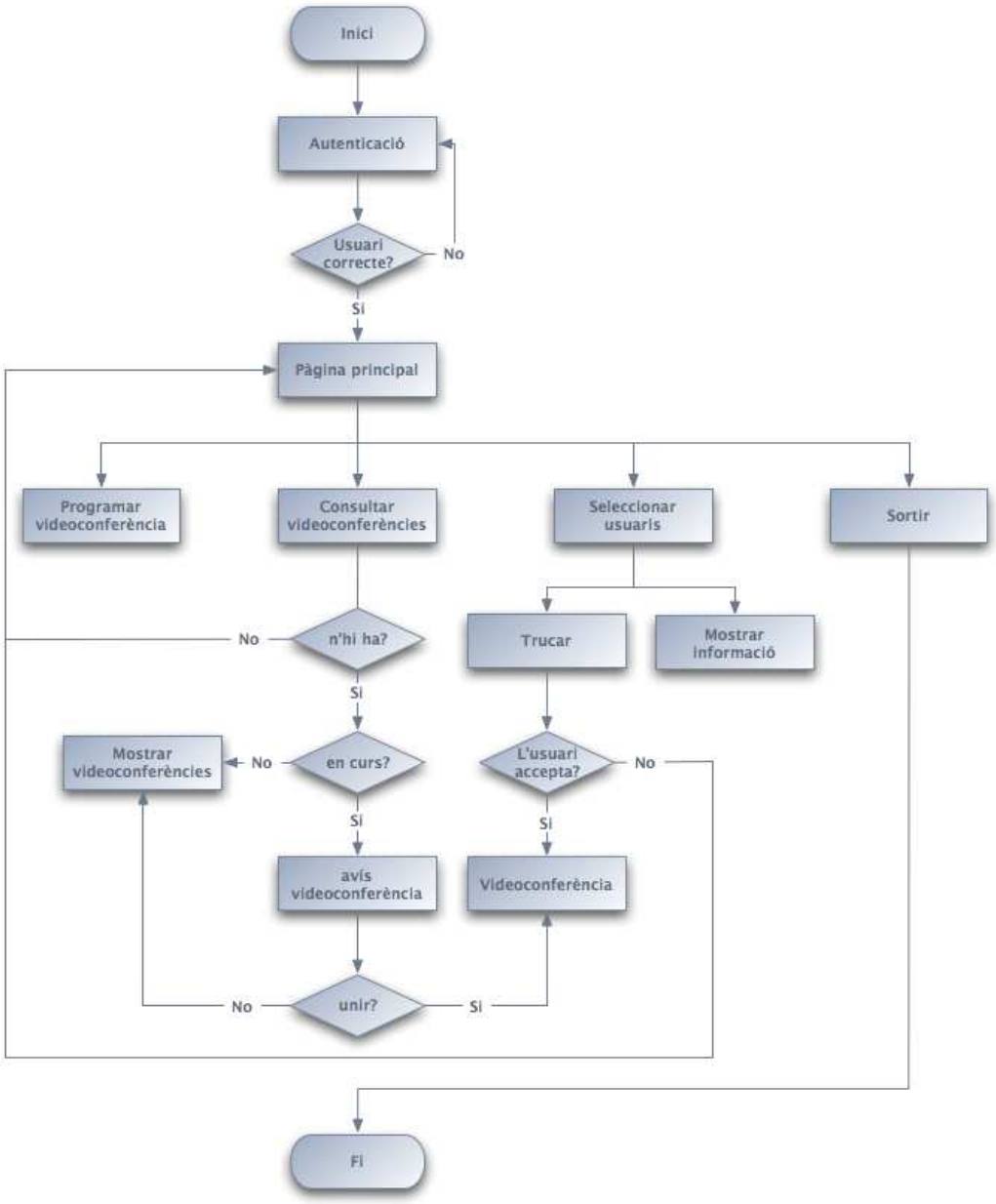


Fig. A.1 Diagrama de flux de l'aplicació WiCat

Aquest diagrama mostra la lògica que segueix el client. Es pot observar que un cop l'usuari s'ha registrat té vàries opcions:

- **Programar videoconferència**: En aquest punt l'usuari ha de seleccionar la data, l'hora, els usuaris implicats i la descripció d'aquesta videoconferència. L'aplicació Flash enviarà aquesta informació al WiCat Server i al Gestor de Videoconferències l'emmagatzemarà.

- **Consultar videoconferències**: És possible que no existeixi cap videoconferència programada en la qual l'usuari estigui convidat. Si n'existeix alguna és possible que estigui en curs, és a dir, que s'estigui realitzant en aquell moment i l'usuari hagi fet tard. Si aquest és el cas, l'usuari té la opció d'unir-se a la videoconferència. Si no és el cas, es mostrarà la llista de videoconferències a les que està convidat.

- **Seleccionar usuaris**: Es poden seleccionar un o varis usuaris de la llista de presència, un cop seleccionats hi ha dues opcions:

    - **Trucar**: Si els usuaris seleccionats estan connectats se'ls pot trucar. Si un o més d'un accepta la trucada s'estableix la videoconferència.

    - **Mostrar informació**: Amb aquesta opció es pot veure la "fitxa" de cada usuari seleccionat amb la seva informació de contacte.

- **Sortir**: Si es prem el botó de sortir es desconnecta l'aplicació, i es retorna a la pàgina de benvinguda, si vol tornar a entrar l'usuari s'haurà d'autenticar altre cop.

A.2. Captures de pantalla

Per implementar les funcionalitats del WiCat Client s'han utilitzat varis arxius SWF. D'aquesta manera la programació ha sigut modular i la integració posterior més senzilla. Flash permet definir una àrea de l'escenari on es poden carregar diferents pel·lícules (arxius SWF) segons convingui.

S'utilitza una pel·lícula principal que s'executa contínuament. Aquesta és l'encarregada de comunicar-se amb el WiCat Server i de carregar els arxius SWF indicats segons els missatges rebuts del servidor o les opcions que esculli l'usuari.

Tot seguit s'enumeren els diferents arxius SWF utilitzats amb les funcionalitats que implementa cadascun.

A.2.1. UserLogin.swf

Aquest arxiu s'encarrega d'obtenir el nom d'usuari i la contrasenya.
En la captura s'observen les diferents parts d'aquest panell.

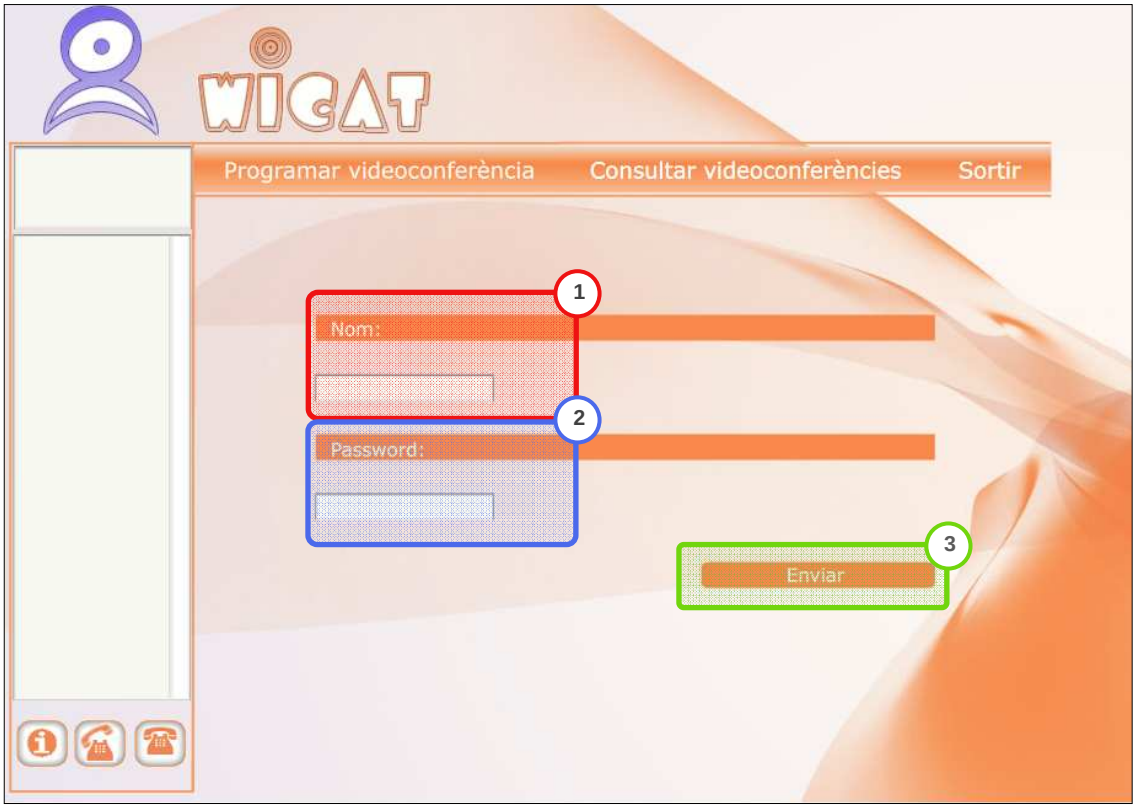


Fig. A.2 Panell d'Autenticació

1- Quadre de text per introduir el nom d'usuari
2- Quadre de text per introduir la contrasenya
3- Botó per enviar les dades al WiCat Server

Si l'usuari no introdueix algun dels dos camps requerits es mostra un missatge d'error. Si l'usuari no és correcte, se'n mostra un altre. No podrà utilitzar la web fins que introdueixi un nom d'usuari existent en el directori LDAP.

A.2.2. Home.swf

Aquest arxiu es carrega un cop l'usuari s'ha autenticat correctament, també quan l'usuari acaba una videoconferència, una programació o una consulta.
Per poder utilitzar correctament l'aplicació conté una petita guia d'ús.

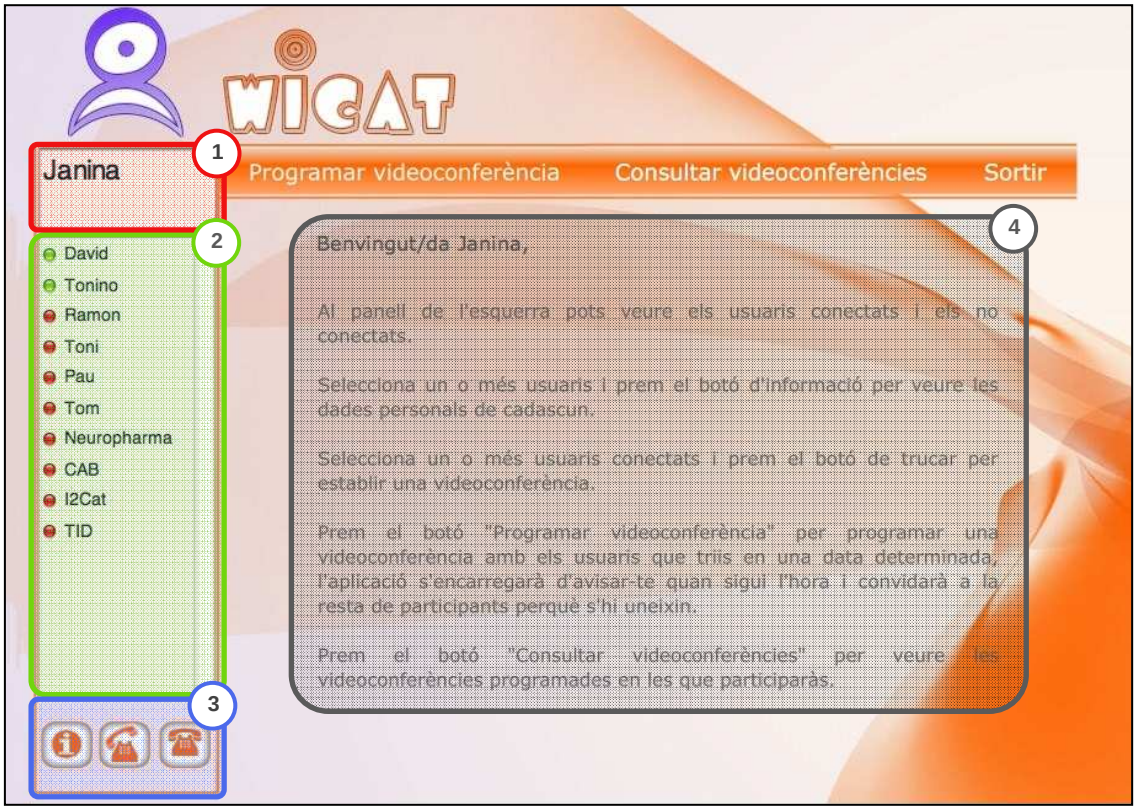


Fig. A.3 Home

1- Panell informatiu. En funcionament normal mostra el nom de l'usuari autenticat.
2- Llista de contactes. S'observen els usuaris connectats i no connectats.
3- Panell d'accions. Inclou el botó d'informació d'usuaris, el de trucar i el de penjar.
4- Panell principal. Mostra una guia d'us. Quan és l'hora es carreguen els arxius SWF corresponents.

A.2.3. ProgConf.swf

ProgConf es carrega quan l'usuari prem el botó "Programar videoconferència".

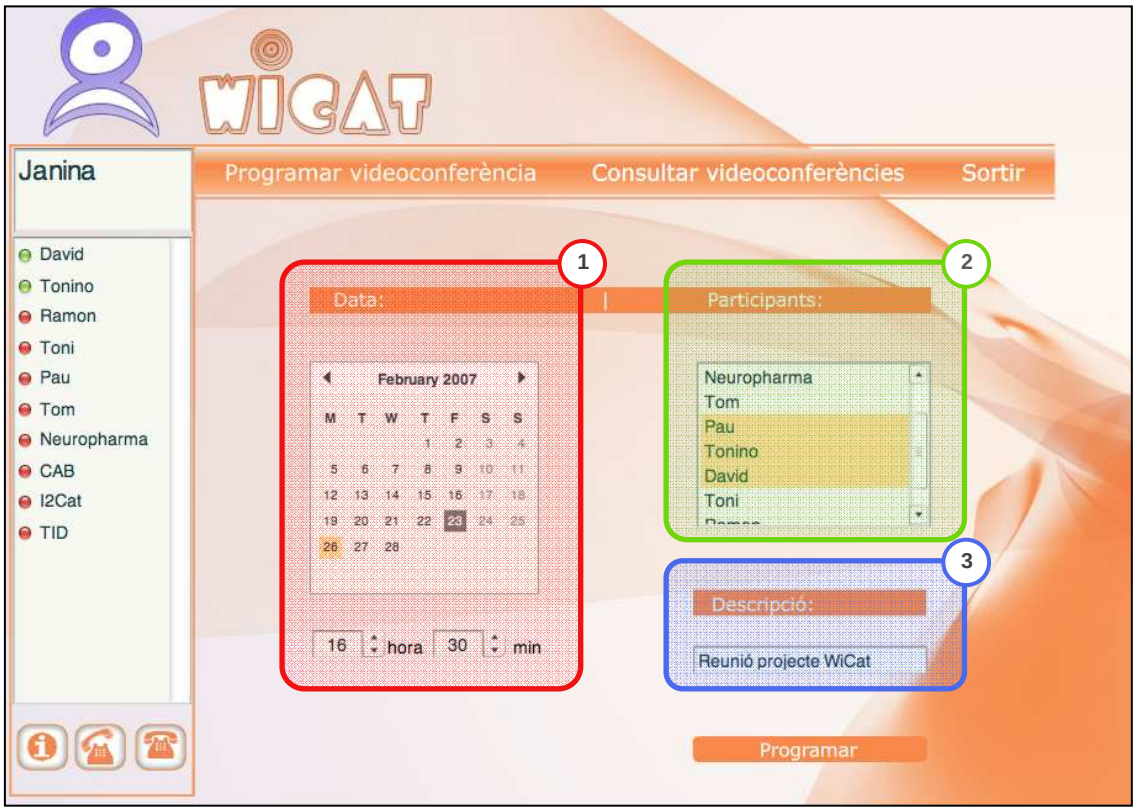


Fig. A.4 Panell de programació de videoconferències

1- Selector de data, hora i minut.
2- Selector d'usuaris participants en la videoconferència
3- Quadre de text per introduir la descripció de la videoconferència.

Un cop introduïts tots els camps (si no s'introdueixen tots, l'aplicació els demana) prement el botó "programar" s'envia la informació cap al WiCat Server que l'emmagatzema en el gestor de videoconferències.

A.2.4. ConsultConf.swf

Aquest arxiu es carrega quan l'usuari prem el botó "Consultar videoconferències". Un cop carregat, si existeixen conferències programades on l'usuari estigui convidat, prement el botó "consultar" es mostren. En cas contrari es mostra un missatge informant de l'absència de videoconferències.

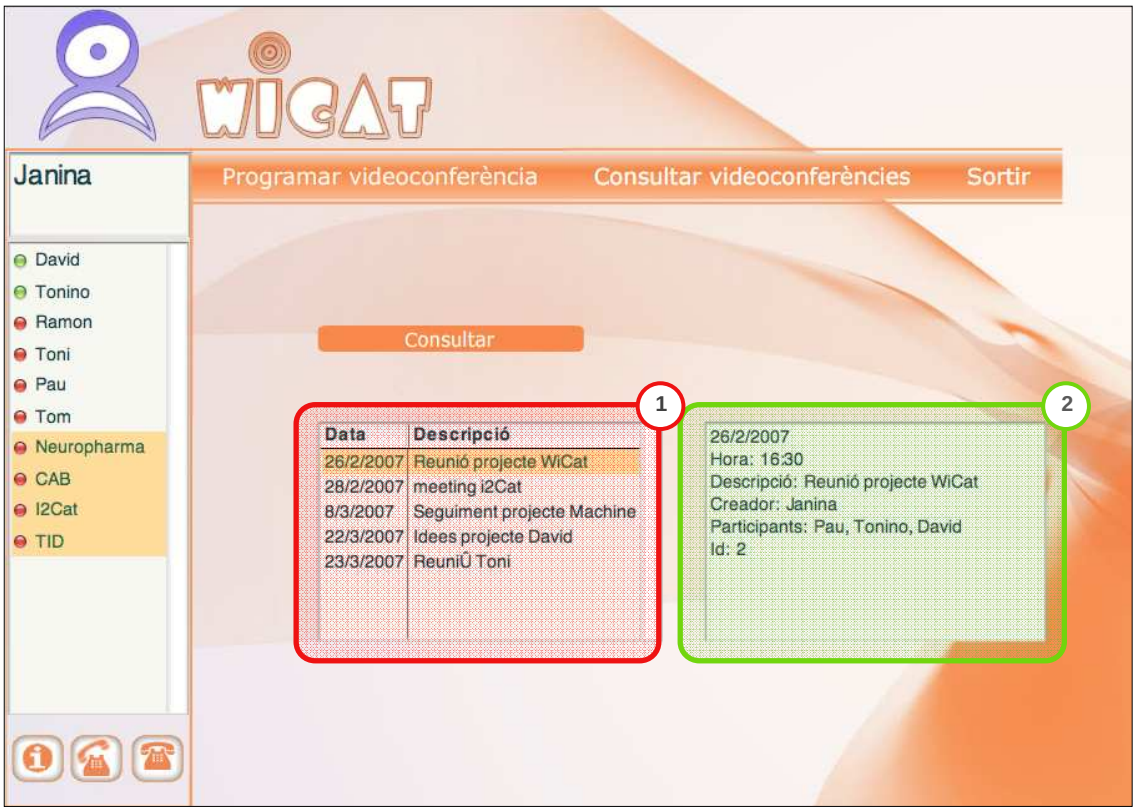


Fig. A.5 Panell de consulta de videoconferències

1- Component DataGrid on es mostra la data de la videoconferència a la primera columna i la descripció en la segona.
2- Llista de les característiques de la videoconferència seleccionada.

A.2.5. ContactInfo.swf

ContactInfo.swf s'utilitza per mostrar la informació de contacte dels usuaris, disponible en el directori LDAP. Per carregar-lo l'usuari ha de seleccionar un o més usuaris de la llista de contactes i prémer el botó d'informació situat just a sota.

Fig. A.6 Panell d'informació d'usuaris

1- Usuaris seleccionats de la llista de contactes.
2- Component *Accordion* que mostra la informació de cada usuari quan es selecciona.

A.2.6. Call.swf

Aquest és l'arxiu més important. Es carrega quan l'usuari inicia una trucada o quan accepta una trucada entrant. Si l'usuari rep una invitació a una multi conferència, abans de carregar l'arxiu Call.swf ha d'acceptar la trucada entrant. Per avisar a l'usuari d'una trucada sona un timbre i apareix el següent avís.

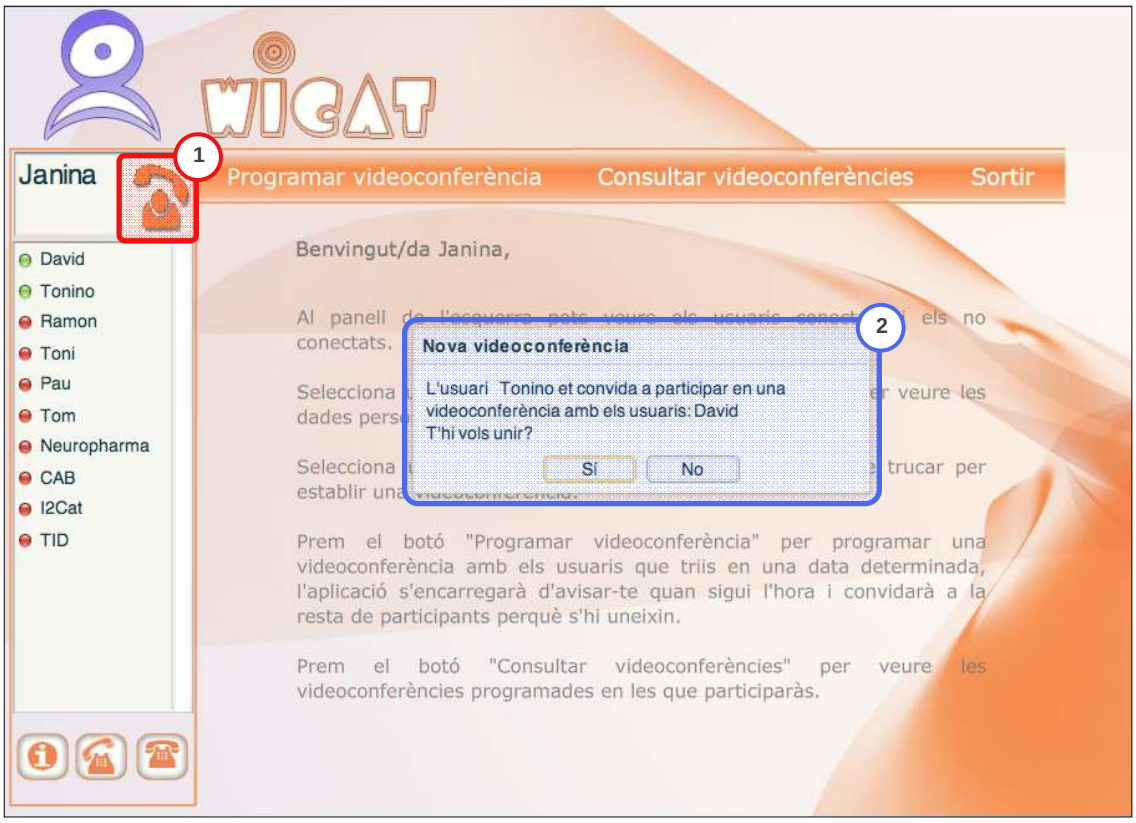


Fig. A.7 Avís de videoconferència

1- Telèfon que es mou d'un costat a l'altre.
2- Finestra d'avís on s'informa a de l'usuari que inicia la trucada i de la resta d'usuaris convidats (en cas que existeixin).

Si l'usuari prem el botó "Si" es carregarà l'arxiu Call.swf amb el nombre indicat d'objectes vídeo. En aquest cas la videoconferència és de tres usuaris.

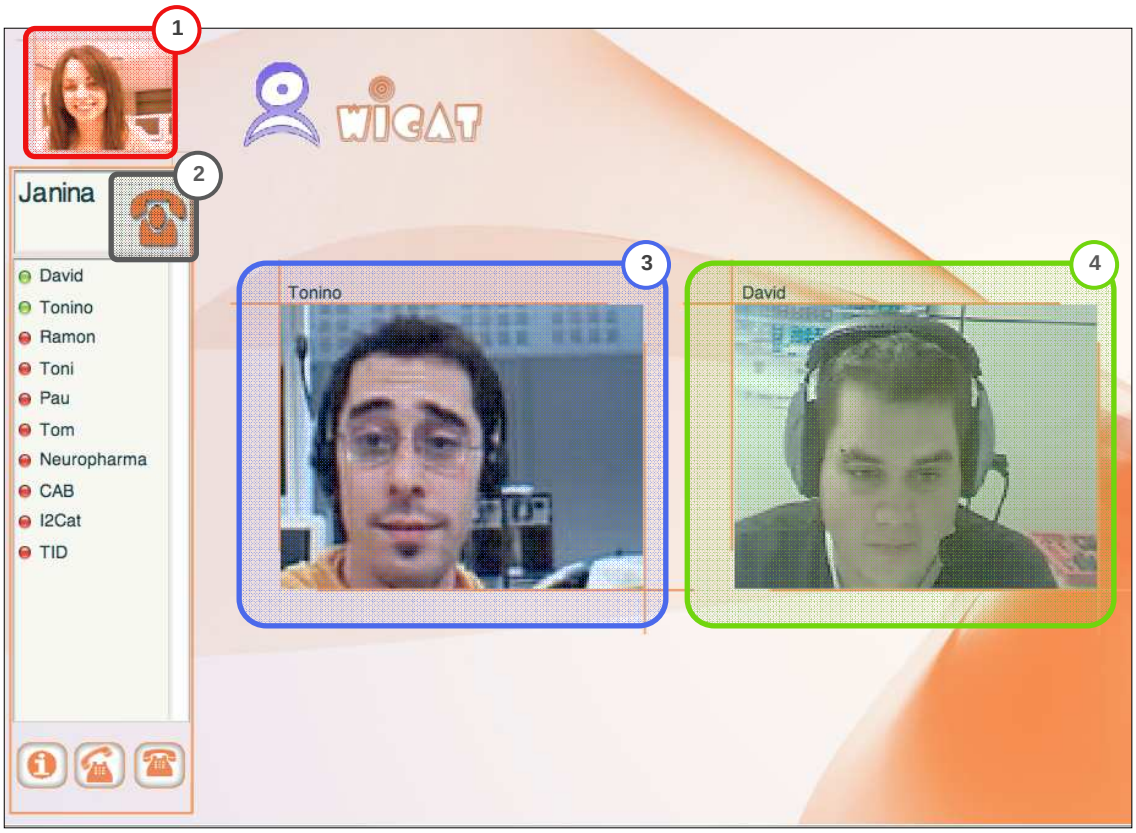


Fig. A.8 Panell de videoconferència

1- Vídeo de l'usuari local.
2- Telèfon estàtic que indica la realització d'una trucada.
3,4- Vídeo dels usuaris remots

ANNEX B. TUTORIAL DEL FLASH MEDIA SERVER 2

La finalitat d'aquest tutorial és guiar a l'usuari en la instal·lació, manteniment i desenvolupament d'aplicacions amb l'Adobe Flash Media Server2.

B.1. Instal·lació del Flash Media Server2

B.1.1. Requeriments del sistema

Requeriments de software:

- La plataforma Flash Media Server2.

  En el següent link es pot trobar la versió gratuïta per desenvolupadors.
  http://www.adobe.com/es/products/flashmediaserver/

- L'entorn de desenvolupament Macromedia Flash 8.

  Es pot trobar una versió d'avaluació de trenta dies en el següent link:
  http://www.adobe.com/es/products/flash/flashpro/

Requeriments de hardware:

- Requeriments mínims de Hardware per instal·lar el Flash Media Server:

  CPU X86-compatible (Pentium lll, 1 GHz o millor).
  512 MB de RAM.
  50 MB d'espai lliure en el disc dur.

- Requeriments recomanats de Hardware per la producció del Flash Media Server:

  CPU X86-compatible (Pentium 4, 3.2 GHz o millor).
  2 GB de RAM.
  200 MB d'espai lliure en el disc dur.
  Targeta Ethernet de 1GB.

B.1.2. Instal·lar el servidor

La instal·lació del Flash Media Server tant en Windows com en Linux és molt senzilla. Durant la instal·lació es demanarà un nom d'usuari i una contrasenya d'administració. Aquestes dades es podran canviar un cop instal·lat el servidor.

No és necessari explicar els passos de la instal·lació en Windows o en Red-Hat ja que són extremadament senzills. Tot i així s'explica com instal·lar-lo en Debian, ja que s'han de fer alguns passos addicionals:

Llibreries necessàries

En Debian es necessiten algunes llibreries, per instal·lar-les s'ha d'utilitzar la següent comanda:

```
$ apt-get install libstdc++2.10-glibc2.2 libnspr4
```

Lincar les llibreries

S'han de lincar les llibreries perquè el Flash Media Server pugui trobar-les amb els noms que ell busca:

```
$ cd /usr/lib
$ ln -s libcrypto.so.0.9.7 libcrypto.so.4
$ ln -s libssl.so.0.9.7 libssl.so.4
```

Instal·lar

S'ha de descomprimir l'arxiu .tar.gz i executar el programa d'instal·lació. S'ha de passar un paràmetre perquè ignori el fet que el sistema operatiu no és un Red-Hat.

```
# ./installFMS –platformWarnOnly
```

Les respostes a les preguntes que es fan durant la instal·lació poden ser com les següents:

Installation directory	= /usr/local/macromedia/fms
FM Server Port	= 1935
FM Admin Server Port	= 1111
Administrative username	= admin
Administrative password	= admin
FMS owner	= www-data
FMS service user	= www-data
FMS service user group	= www-data
FMS run as daemon	= Yes
Start FMS	= Yes

En debian, si es vol que el FMS s'executi automàticament després de reiniciar, s'ha d'utilitzar la següent comanda:

```
$ ln -s /etc/init.d/fms /etc/rc2.d/S999fms
```

Per parar el FMS abans de reiniciar o apagar l'equip utilitzar les següents comandes:

```
$ ln -s /etc/init.d/fms /etc/rc6.d/K20fms
$ ln -s /etc/init.d/fms /etc/rc0.d/K20fms
```

B.1.3. Iniciar el servidor

Windows:

Per iniciar el servidor manualment, seleccionar menú Inici > Programes > Macromedia > Flash Media Server 2 > Start Flash Media Server i Start Flash Media Admin Server.

Linux:

Per iniciar el servidor manualment, s'ha d'executar la següent comanda:

```
$ fmsmgr server start
```

El servei ja està arrancat, a partir d'aquí aquest tutorial conté les següents seccions:

- Creació d'aplicacions amb el Flash Media Server 2
- Administració del Flash Media Server 2
- Seguretat en el Flash Media Server 2

B.2. Creació d'aplicacions amb el Flash Media Server 2

B.2.1. Creació de l'entorn de desenvolupament

Abans de començar el desenvolupament d'una aplicació, s'ha de preparar l'entorn. Per això s'han de realitzar una sèrie d'accions:

Especificar la URI del servidor.[1]

- Si s'executa en la mateixa màquina que l'entorn de desenvolupament Macromedia Flash s'ha d'utilitzar una línia de codi com la següent:

  ```
  new_nc.connect("rtmp:/aplicacio/instancia");
  ```

- Si aquest no és el cas i el servidor s'executa en una altra màquina com per exemple a *servidor.domini.com* s'ha d'utilitzar la següent línia:

  ```
  new_nc.connect("rtmp://domini.com/aplicacio/instancia");
  ```

Escriure el codi Action Script del client. El codi del client Flash s'ha d'adjuntar a una capa en el primer *Frame* del arxiu FLA, mai en objectes individuals.

[1] S'han d'utilitzar les dos barres (//) després d'rtmp: L'ús d'una sola barra és possible únicament quan l'aplicació SWF es serveix des de la mateixa màquina on s'executa el Flash Media Server.

Escriure el codi Action Script del servidor. Per les aplicacions que utilitzin Action Script de servidor, s'ha d'escriure el codi en un script de servidor. Aquest script es pot anomenar *main.asc* o *nom_de_aplicacio.asc*. Es pot utilitzar el Flash Action Script Editor o un editor de JavaScript per escriure'l.

Carregar el components.asc. Per utlilitzar components de media en una aplicació, s'ha de carregar el script components.asc, localitzat en el directori scriptlib. Per carregar aquest arxiu, s'ha de crear un script de servidor i afegir la següent línia al principi del codi:

```
load("components.asc");
```

Monitoritzar les aplicacions. Mentre l'aplicació està en fase de proves es poden mostrar els logs que genera, els valors dels objectes compartits,etc. Per fer això s'utilitza la consola d'administració.

B.2.2. Desenvolupar aplicacions amb el Flash Media Server

Per desenvolupar una aplicació amb el Flash Media Server s'han de seguir els següents passos:

1.- Triar un nom per l'aplicació, per exemple *my_app* i registrar l'aplicació en el servidor. Per això s'ha de crear un directori amb el nom de l'aplicació i posar-lo en el directori d'aplicacions del Flash Media Server: En Windows *C:\Archivos de programa\Macromedia\Flash Media Server 2\Applications*. En Unix */opt/macromedia/fms/applications*.

2.- Per la part del client, s'ha de crear un arxiu FLA mitjançant l'eina de desenvolupament de Macromedia Flash. Aquest arxiu ha d'incloure un objecte *NetConnection* i connectar-lo a l'aplicació:

```
nc = new NetConnection();
nc.connect("rtmp://domini.com/aplicacio");
```

3.- Guardar l'arxiu FLA. Aquest arxiu no forma part de l'aplicació desenvolupada i es pot guardar on es vulgui. Es fa servir per generar l'arxiu SWF que els clients utilitzaran per connectar-se al servidor.

4.- Si existeix un fitxer d'Action Script de servidor s'ha de posar en el directori de l'aplicació registrada en el Flash Media Server o en un directori *scripts* dins del mateix. Aquest script es pot dir *main.asc* o *nom_aplic_registrada.asc*.[1]

5.- Publicar l'arxiu SWF per l'aplicació en un directori accessible pels clients però que no utilitzi el Flash Media Server. Per exemple en el directori de publicació d'un servidor web juntament amb els arxius HTML i Action Script de client (.as) que utilitzi l'aplicació client. Els arxius FLA poden estar junt amb els

[1] Encara que l'aplicació del servidor no contingui cap script s'ha de crear el directori amb el nom de l'aplicació. El Flash Media Server guardarà en aquest directori qualsevol flux o objecte compartit que crei l'aplicació.

SWF mentre l'aplicació està en fase de desenvolupament, un cop en producció, s'han d'esborrar del directori de publicació i guardar-los en un lloc segur.

B.2.3. Instàncies d'aplicacions

Les aplicacions s'executen creant instàncies d'aplicació. Quan un client es connecta a una aplicació, en realitat s'està connectant a una instància d'aquesta aplicació. Per exemple, un client es connecta a una aplicació anomenada *chat*:

```
nc.connect("rtmp://domini.com/chat");
```

En l'exemple el client no especifica la instància de l'aplicació a la que es vol connectar, per tant, es connecta a la instància per defecte (_defInst_).

Els clients es poden connectar a una instància concreta de l'aplicació especificant-la en la URI:

```
nc.connect("rtmp://domini.com/chat/instancia1");
```

Normalment s'utilitzen instàncies d'aplicacions per propòsits concrets. Per exemple pot ser necessari tenir diferents "sales" en una aplicació perquè els clients que es connectin a una sala interactuïn només entre ells.

```
nc.connect("rtmp://domini.com/chat/sala1");
nc.connect("rtmp://domini.com/chat/sala2");
```

Una instància d'aplicació no necessita el seu propi directori definit en el servidor. Tot i això els recursos de cada instància (fluxos i objectes compartits) són independents i s'emmagatzemen en directoris propis que es creen dins dels directoris definits pels fluxos i objectes compartits de l'aplicació.

B.2.4. Exemple d'aplicació

Per la realització del TFC s'ha utilitzat una aplicació senzilla per poder compartir fluxos entre diferents usuaris. La lògica s'efectua en la part del client, el script *main.asc* de servidor utilitzat és molt senzill, s'encarrega d'acceptar les connexions dels clients.

```
load("components.asc");

application.onAppStart = function()
{
          this.uniqueID = 0;
          trace("APP START!!")
};
application.onConnect = function(newClient)
{
          trace("new client!!");
          newClient.id = this.uniqueID++;
          this.acceptConnection(newClient);
          trace("connection accepted");
};

application.onDisConnect = function(client)
{
          trace("APP STOP!!")
};
```

Els clients, per la seva banda s'encarreguen de publicar els seus fluxos d'àudio i vídeo i de subscriure's als fluxos dels altres clients que participen en la videoconferència. En l'exemple es pot veure com l'usuari anomenat Joan publica el seu flux i es subscriu al de l'usuari Ramon.

```
// captura el flux d'àudio i vídeo de la càmera i el micròfon
var cam:Camera = Camera.get();
var mic = Microphone.get();

// crea la variable NetConnection i es connecta al Flash Media Server
var nc:NetConnection = new NetConnection();
nc.connect("rtmp://servidor.com/aplicacio");

// crea la variable NetStream i l'associa a la NetConnection
var ns:NetStream = new NetStream(nc);

// publica el flux d'àudio i vídeo amb l'identificador Joan
ns.attachVideo(cam);
ns.attachAudio(mic);
ns.publish("Joan");

// crea un objecte vídeo per reproduïr el flux de l'usuari Ramon
var ext_video:Video = new Video();

// crea un altre objecte NetStream per obtenir el flux del Ramon
var ext_ns:NetStream = new NetStream(nc);
ext_ns.play("Ramon");

// reprodueix el vídeo de l'usuari Ramon en l'objecte Video
ext_video.attachVideo(ext_ns)
```

L'aplicació del Flash Media Server s'encarrega d'obtenir els fluxos que publiquen els usuaris i d'emmagatzemar-los en el directori registrat. D'aquesta manera els usuaris que es subscriguin a aquests fluxos els podran obtenir i reproduir.

B.3. Administració del Flash Media Server 2

El Flash Media Server 2 disposa d'una consola d'administració per web on es poden administrar els servidors que s'executen en el Flash Media Server, monitoritzar els seus processos i *debuguejar* les seves aplicacions.

B.3.1. Connectar amb la consola d'administració

Per connectar amb la consola d'administració s'han de seguir els següents passos:

1.- En Windows, des del menú inici, seleccionar Programes->Macromedia->Flash Media Server->Console. En Linux, obrir el fitxer fmsconsole.html en un navegador web d'un ordinador que tingui instal·lat el Flash Player.

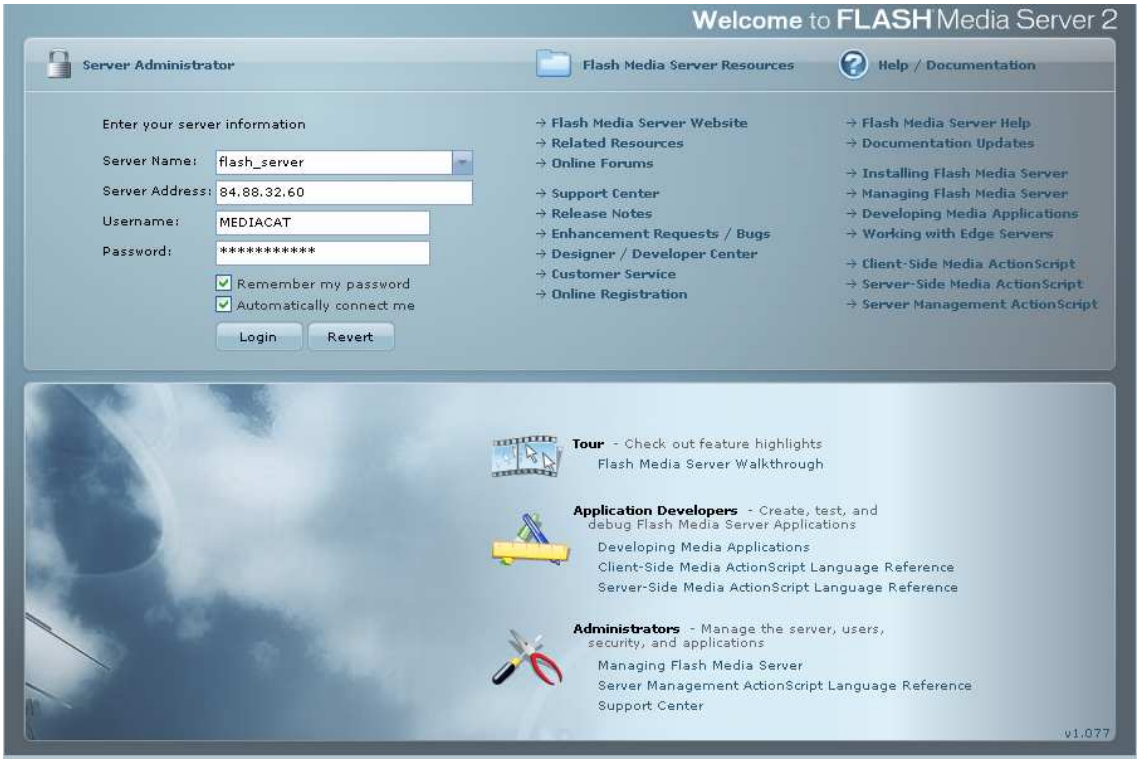


Fig. B.1 Consola d'administració

2.- Escriure el nom i adreça del servidor al que es vol connectar.

- Per referir-se al mateix ordinador on s'executa la consola d'administració, es pot introduir "localhost".

- Si la consola d'administració s'executa en un altre ordinador, s'ha d'introduir el nom del servidor (FMS.myCompany.com) o l'adreça IP i el port on es vol connectar (84.88.32.33:1111). Per defecte el port d'administració del Flash Media Server és el 1111.

3.- Introduir el nom d'usuari i el password d'administració i clicar el botó de Login. Aquest nom i password corresponen amb els introduïts durant la instal·lació del Flash Media Server. Es poden canviar en qualsevol moment mitjançant la consola d'administració o l'arxiu Users.xml.

La connexió amb el Flash Media Server està establerta. En les següents seccions es poden veure les diferents accions que es poden realitzar amb la consola d'administració:

- Administrar aplicacions
- Administrar usuaris
- Administrar servidors

B.3.2. Administrar aplicacions

El panell "View Applications" mostra informació sobre les aplicacions que s'executen en el servidor. Des d'aquí l'administrador pot monitoritzar l'estat d'una aplicació.

Crear una instància d'aplicació

En aquesta secció es pot crear una nova instància d'aplicació clicant el botó "New Instance".

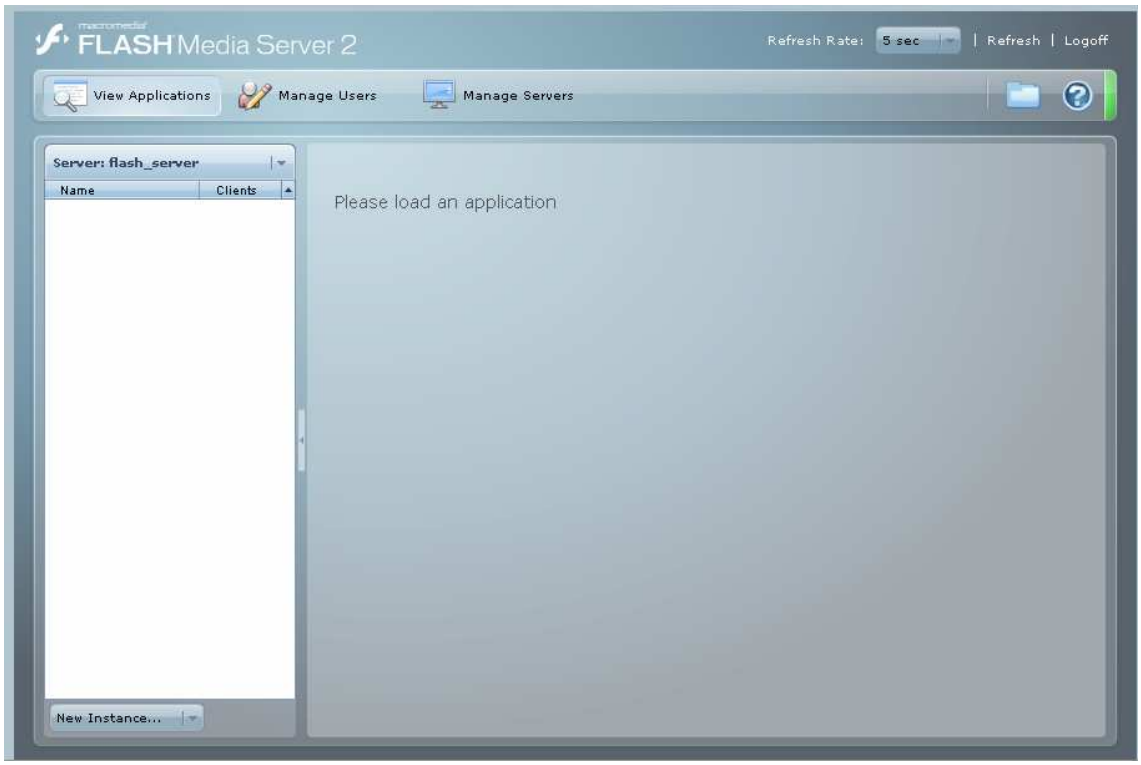


Fig. B.2 Panell "View Applications"

Aquesta acció crea una instància d'aplicació i l'afegeix a la llista. Per defecte el nom de la instància és *_defInst_*, però aquest valor es pot editar si es desitja tenir varies instàncies per una aplicació.

Un cop creada la instància, la consola d'administració permet visualitzar vaires opcions.

Veure els logs de l'aplicació en viu

Cada aplicació té associat un fitxer de log on guardarà els missatges generats durant l'execució. Aquest fitxer es pot visualitzar dinàmicament en aquest panell.

És possible buscar *strings* en els missatges de log mitjançant l'eina de cerca. El botó "clear log" neteja la finestra de logs.

Veure els clients actius

En el panell "Clients" es mostra la llista de clients connectats a l'aplicació. Per cada client hi ha la següent informació:

- Identificador de connexió.
- Protocol de connexió.
- Nombre de bytes rebuts pel client i la informació retornada.
- Temps de connexió.
- Nombre de missatges d'entrada i sortida de l'aplicació.
- Missatges perduts.

Veure els objectes compartits

La pestanya "Shared Objects" mostra un panell amb la llista d'objectes compartits per l'aplicació. Per cada objecte es pot veure el seu nom, el tipus (volàtil o persistent), i les connexions (nombre d'usuaris subscrits a aquest objecte compartit). La informació d'aquest panell ajuda a *debuguejar* una aplicació.

Veure els fluxos actius

Aquest panell "Streams" mostra els fluxos actius d'una aplicació. Es poden veure els seus noms i els tipus. Si es selecciona un flux es mostren les seves propietats.

Si la connexió de *debug* està disponible apareix un botó "Play Stream". Prement-lo es reprodueix el flux seleccionat.

Veure el rendiment d'una aplicació

Seleccionant la pestanya "Performance" es mostra la informació de l'aplicació. L'administrador visualitza les següents dades:

- Informació del client: Nombre total de clients, nombre de connexions actives i nombre de connexions rebutjades.
- Temps de vida de l'aplicació: Indica l'hora d'inici de l'aplicació i quant temps fa que s'executa.
- Nombre de missatges d'entrada i sortida de l'aplicació.
- Nombre de bytes entrants i sortints de l'aplicació.

També es mostra informació sobre el Flash Media Server:

- Nombre de connexions actives que suporta.
- L'ample de banda que consumeixen els recursos.
- Percentatge de CPU i memòria que es consumeix.

B.3.3. Administrar usuaris

En aquesta secció es controlen els usuaris administradors del Flash Media Server i els seus permisos. Es poden efectuar les següents accions:

- Afegir un nou usuari administrador.
- Eliminar administradors.
- Canviar les contrasenyes d'administració.

A l'esquerra del panell "Manage Users" es mostren els usuaris. A la dreta s'aporta informació quan es selecciona un usuari.

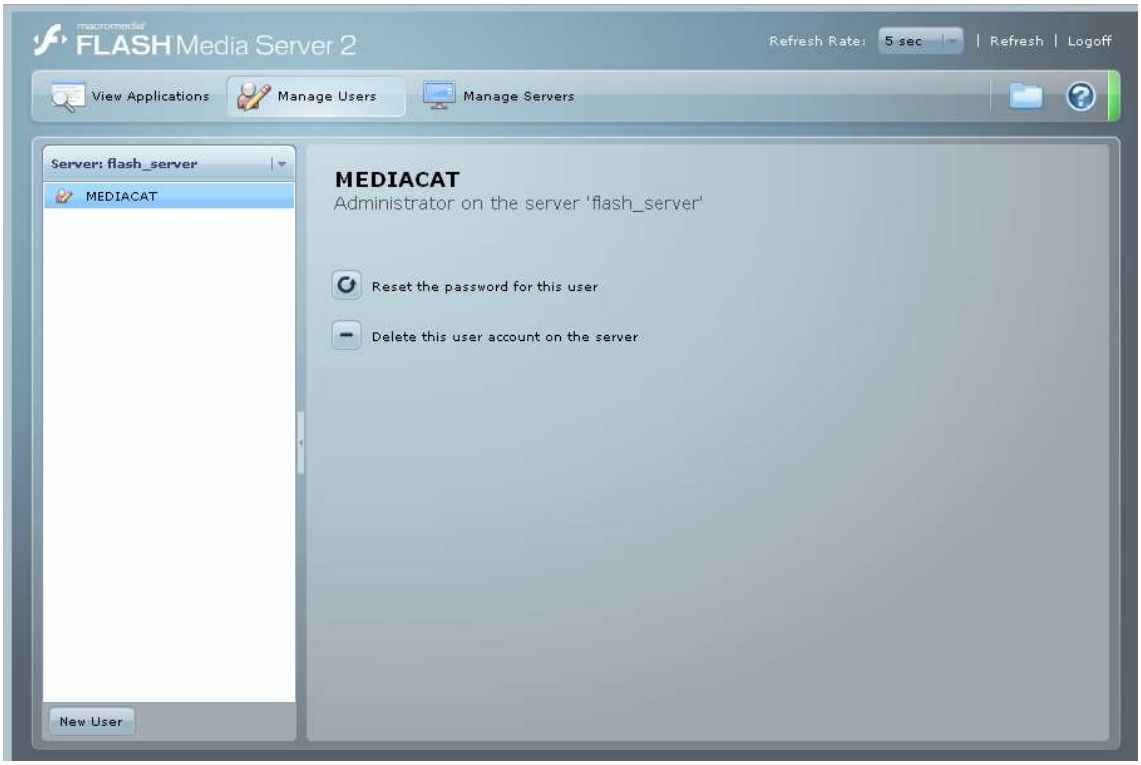
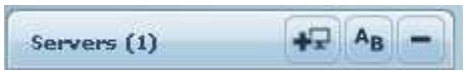


Fig. B.3 Panell "Manage Users"

B.3.4. Administrar servidors

El panell “Manage Servers” conté una llista amb els servidors a l’esquerra. A sobre d’aquesta llista es poden veure una sèrie de botons que efectuen les següents accions:



- Afegir un servidor a la llista d’administració.
- Editar la informació de “Login” per un servidor.
- Eliminar un servidor de la llista.

A sota de la llista hi ha una sèrie de botons que permeten executar accions sobre un servidor seleccionat:

- Connectar la consola d’administració a un servidor seleccionat. La consola es pot connectar a varis servidors simultàniament.
- Fer un ping al servidor per verificar que s’està executant i veure el seu temps de resposta en milisegons.
- Reiniciar (o iniciar) el servidor seleccionat.
- Veure els servidors eliminats.
- Parar el servidor seleccionat.

A la dreta d’aquest panell es poden seleccionar diferents pestanyes per administrar cada servidor:

Veure els detalls del servidor

La pestanya “Details” obre un panell on es mostren els clients actius i el temps de vida del servidor. A més es poden veure tres gràfiques que mostren les connexions actives, els kilobits entrants i sortints, i els percentatges de memòria i CPU utilitzats.

Veure les connexions

La secció “Connections” mostra informació sobre les connexions a un cert servidor. Indica les connexions actives, les totals, els kilobits entrants i sortints, i els missatges.

Veure les aplicacions

La pestanya “Applications” mostra informació de totes les aplicacions que s’executen en el servidor. Aporta la següent informació: Nom de l’aplicació, nombre d’instàncies carregades de l’aplicació, nombre d’usuaris que estan connectats, i nombre total de connexions acceptades i rebutjades per cada aplicació.

Veure la llicència del Flash Media Server

El panell “Licence” mostra la informació detallada de la llicència del Flash Media Server. Es pot seleccionar una llicència individual per veure’n els detalls. Per cada “serial key” es mostra la següent informació:

- Nombre de connexions autoritzades.
- Màxim ample de banda permès.
- Data de caducitat de la llicència, si escau.

Pel cas de la versió gratuïta per desenvolupadors del Flash Media Server només es limiten el nombre de connexions, concretament a deu.

Veure els logs del servidor

La pestanya “Server Log” mostra un panell amb traces emmagatzemades en el fitxer de log del servidor.

B.4. Seguretat en el Flash Media Server 2

B.4.1. Protegir el Flash Media Server amb un firewall

És recomanable instal·lar el Flash Media Server en una zona desmilitaritzada d’una xarxa (DMZ). Aquesta configuració consta d’un firewall que protegeix els equips de la xarxa d’atacs des de l’exterior i als equips de la DMZ d’atacs tant de l’interior com de l’exterior de la xarxa privada. Les connexions cap a la IP pública d’aquesta xarxa es redireccionen cap a la DMZ, d’aquesta manera els clients que vulguin connectar-se al Flash Media Server podran fer-ho mitjançant la IP pública i la xarxa local estarà protegida.

El Flash Media Server rep connexions al port 1935 per defecte, si només es disposa d’aquest servidor en la DMZ s’haurà d’obrir només el port 1935 del firewall. Si a més del FMS es disposa d’un altre servidor, com ara un servidor web, en el firewall també s’obrirà el port 80 per permetre connexions HTTP.

B.4.2. Controlar connexions des de l’interior d’un tallafocs

El Flash Media Server no realitza cap connexió amb els clients Flash (arxius SWF). El Flash sempre inicia les connexions des del client. És possible que els usuaris es trobin en l’interior d’una xarxa protegida per un tallafocs. Si aquest tallafocs permet als ordinadors de la xarxa privada connectar-se a qualsevol port de qualsevol servei de l’exterior del tallafocs, no hi haurà cap problema.

- El navegador web de l’usuari farà una petició HTTP al servidor web i aquest li retornarà l’arxiu SWF.
- L’arxiu SWF s’executarà en el navegador web del client i es connectarà al Flash Media Server en el port 1935.

En alguns casos els tallafocs d'algunes organitzacions bloquegen tots els ports excepte els que fan servir els navegadors web. Normalment s'utilitzen el port 80 per HTTP i el 443 per HTTPS. Sovint els servidors web fan servir el 8080 com a port secundari per l'HTTP.

Fig. B.4 Tallafocs bloquejant tots els ports excepte el 80

En la imatge s'observa que els clients no poden connectar-se al Flash Media Server perquè el tallafocs bloqueja la connexió.

El Flash Media Server es pot configurar perquè accepti connexions en altres ports apart del 1935, inclús en més d'un port al mateix temps. Per exemple, es pot configurar perquè accepti connexions en els ports 1935, 80 i 443. Aquesta opció es defineix en el tag *HostPort* del fitxer *Adaptors.xml*.

```
<HostPort>:1935,80,443</HostPort>
```

Si existeix un servei executant-se en el mateix servidor, a la mateixa IP, no s'ha de configurar el Flash Media Server perquè escolti en el mateix port. Per exemple, si un servidor web utilitza els ports 80 i 443, el Flash Media Server no pot fer servir aquests ports. Amb la configuració de ports de l'exemple poden passar dues coses:

Si no existeix cap servidor web en la mateixa IP:

El client Flash intentarà realitzar una connexió RTMP al port 1935. Com que el seu firewall impedeix les connexions a aquest port, fallarà. Llavors el SWF provarà amb el 443 i si falla, finalment realitzarà la connexió al port 80. Amb la següent línia n'hi ha prou:

```
nc.connect("rtmp://domini.com/aplicacio");
```

Per tant, no és necessari especificar un port en la URI RTMP.

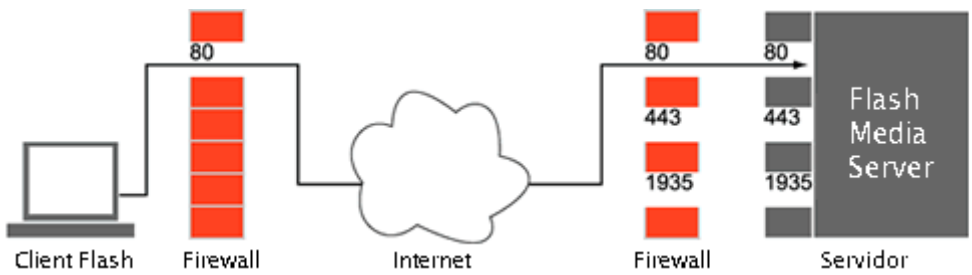


Fig. B.5 Connexió al port 80 del FMS

Si existeix un servidor web en la mateixa IP:

El Flash Media Server no pot escoltar pel port 80 ni pel 443, per tant el client bloquejat per un firewall no podrà establir una connexió RTMP amb el Flash Media Server. Per solucionar aquest problema es proposen varies opcions:

- No posar un servidor web en la mateixa màquina on s'executa el FMS. Aquesta opció, encara que no sempre és possible, és la millor idea. No existeix cap restricció que impedeixi als arxius SWF connectar-se mitjançant RTMP a un servidor amb diferent adreça IP que el servidor web d'on s'han descarregat.

- Configurar la màquina utilitzada perquè escolti en dues IP's diferents. Es poden utilitzar dues targetes de xarxa, o en alguns casos és possible tenir dues IP's associades a la mateixa interfície de xarxa. Llavors es pot configurar el servidor web perquè escolti en una IP i el FMS en l'altra.

- Configurar el servidor web perquè permeti connexions només en el port 8080 i en el 443, deixant el port 80 lliure perquè l'utilitzi el Flash Media Server.

La última opció és també aplicable al FMS, és a dir, deixar que el servidor web utilitzi el 80 i el 443 i configurar el FMS perquè utilitzi només el 8080. Això pot ser un inconvenient ja que alguns tallafocs bloquegen les connexions al port 8080 i, per tant, un dels dos serveis no seria accessible.

Suposant que és el Flash Media Server qui escolta en el port 8080, existeix un altre inconvenient. S'hauria d'especificar explícitament amb ActionScript de client que l'arxiu SWF s'ha de connectar al port 8080. Si s'especifica el port en la URI, com en el cas següent, el SWF es connectarà directament al port 8080 sense intentar connexions en cap altre port:

```
nc.connect("rtmp://domini.com:8080/aplicacio");
```

Una altra opció és fer una connexió normal, sense especificar el port de connexió. Quan el client Flash hagi fallat intentant la connexió amb els ports 1935, 443 i 80, es pot capturar l'event de fallida mitjançant el *handler* `netConnection.onStatus` i només llavors, fer la connexió al port 8080.

B.4.3. HTTP tunneling

Si els tallafocs només bloquegen les connexions basant-se en l'adreça IP i el port, és possible utilitzar les configuracions anomenades anteriorment. Es pot utilitzar el port 80 per les connexions amb el Flash Media Server.

Existeixen tallafocs de nivell d'aplicació que examinen les dades de l'interior dels paquets que travessen la xarxa, per comprovar si concorden amb el

protocol associat normalment a cert port. Pel cas del port 80, aquests tallafocs buscaran capçaleres HTTP i transaccions HTTP. Quan s'intenta una connexió RTMP al port 80, aquesta no s'assembla a una transacció HTTP. Un tallafocs de la capa d'aplicació ho detectarà i denegarà la petició de connexió.

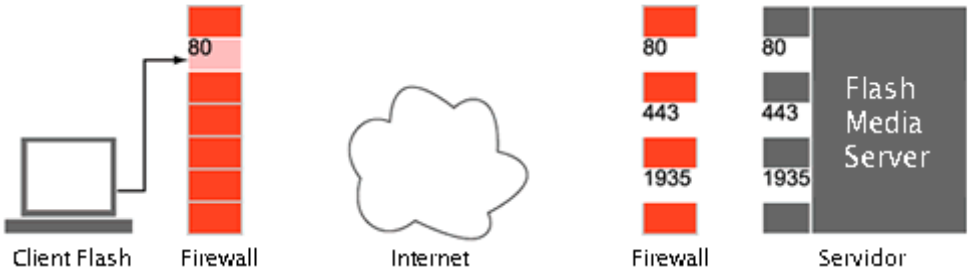


Fig. B.6 Tallafocs de capa d'aplicació

Aquest problema es pot solucionar mitjançant l'HTTP *tunneling*. Aquesta tècnica permet al Macromedia Flash Player i al Flash Media Server transmetre dades RMTP, com àudio i vídeo, embolcallant la informació RMTP dins de peticions HTTP.

Per mantenir les connexions obertes contínuament, com les que permet l'RTMP, el player consulta constantment el servidor mitjançant peticions HTTP.

Aquest mecanisme és menys eficient que l'RTMP estàndard, a causa de les peticions constants i l'embolcallament de les dades RTMP. És recomanable fer servir HTTP *tunneling* només quan és necessari, primer s'ha d'intentar l'ús d'una connexió RTMP estàndard. L'HTTP *tunneling* per l'RTMP s'anomena RTMPT. Un exemple de connexió RTMPT és el següent:

```
nc.connect("rtmpt://domini.com/aplicacio");
```

Si no s'especifica el port, el SWF intentarà la connexió en el port 80 utilitzant HTTP *tunneling*. Per solventar l'ús innecessari d'aquesta tècnica el Flash implementa un quart intent de connexió quan fallen totes les connexions RTMP en tots els ports. Si s'especifica una URI com la següent:

```
nc.connect("rtmp://domini.com/aplicacio");
```

El Flash Player provarà connexions en els següents ports i protocols, en l'ordre llistat:

Taula B.1 Seqüència d'intents de connexió per part del Flash Player

Seqüència	Port	Protocol
1	1935	RTMP
2	443	RTMP
3	80	RTMP
4	**80**	**RTMPT**

El client Flash farà quatre intents per connectar-se al FMS. Els intents u, dos i tres fent servir el protocol RTMP fallaran. El quart intent funcionarà utilitzant el protocol RTMPT en el port 80.

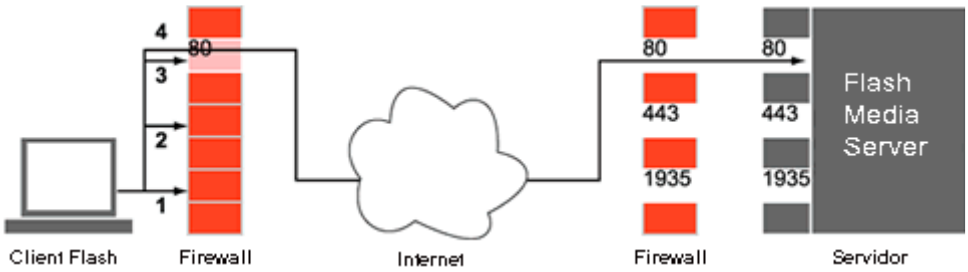


Fig. B.7 Intents de connexió al FMS

B.4.4. Suport d'SSL amb el Flash Media Server

L'SSL (*Secure Sockets Layer*) és un protocol que permet connexions segures mitjançant TCP/IP. El Flash Media Server té suport per connexions SSL. Per realitzar una connexió segura amb el FMS s'utilitza el protocol RTMPS en la URI de connexió:

```
nc.connect("rtmps://domini.com/aplicacio");
```

Per generar els certificats requerits, el Flash Media Server utilitza una llibreria de codi obert, concretament l'OpenSSL. Com que el tamany dels arxius del Flash Player ha de ser el mínim, afegir una llibreria d'SSL en el player no és una opció. Per això el Flash aprofita les llibreries de la plataforma com ara WinINET en Windows per realitzar HTTP tunneling. Una connexió RTMPS entre el Flash Player i el FMS és una connexió HTTP segura (HTTPS).

Per configurar un port com a segur, s'especifica un signe menys abans del número de port en el tag *HostPort* del fitxer *Adaptor.xml*:

```
<HostPort>:1935,80,-443</HostPort>
```

L'exemple especifica que el Flash Media Server escolta en qualsevol interfície de xarxa, en els ports 1935, 80 i 443, on el 443 està designat com a port segur que només rep connexions RTMPS. Amb aquesta configuració, una connexió RTMPS als ports 1935 o 80 fallarà, el client intentarà realitzar un "SSL handshake" que el servidor no serà capaç de completar. Pel mateix motiu, una connexió RTMP al port 443 fallarà, aquest cop serà el servidor el que intentarà un "SSL handshake" que el client no podrà completar. Per mes informació de l'ús d'SSL amb el Flash Media Server consultar la bibliografia [15] .

ANNEX C. TUTORIAL DE SERVEIS WEB

La finalitat d'aquest tutorial és donar unes pautes per poder implementar i desplegar serveis web mitjançant l'eina Apache AXIS.

Entre altres coses l'Axis proporciona:

- Un entorn d'execució per serveis web Java.
- Eines per crear arxius WSDL des de classes Java.
- Eines per desplegar, provar i monitoritzar serveis web.
- Integració amb servidors d'aplicacions i contenidors de Servlets.

C.1. Software necessari

- Màquina virtual de Java versió 1.4 o superior.
- Kit de desenvolupament de serveis web: Apache Axis2
  URL: http://ws.apache.org/axis2/

- Servidor d'aplicacions
  Per exemple: Apache Tomcat versió 4.1 o superior.

C.2. Instal·lació d'Apache Axis

En primer lloc, s'assumeix que el servidor d'aplicacions està escoltant en el port 8080.

- Descarregar els binaris de la distribució d'Axis. Allà s'hi pot trobar la carpeta *axis-versio/webbapps/axis*. Aquesta carpeta s'ha de copiar en la carpeta *webapps* del servidor d'aplicacions (*/tomcat/webapps/axis*).

- Iniciar el servidor d'aplicacions

- Accedir a la pàgina principal:

  http://localhost:8080/axis

- Comprovar que la instal·lació s'ha realitzat correctament clicant en el link *Validate* o accedint a la següent URL:

  http://localhost:8080/axis/happyaxis.jsp

  Aquesta pàgina mostra les llibreries que necessita Axis per funcionar. Si no troba totes les llibreries necessàries no funcionarà. Fins que no es

tinguin totes les llibreries requerides no s'ha de procedir al pas següent de la instal·lació.

Les llibreries s'afegeixen al directori tomcat*/webapps/axis/WEB-INF/lib*.

C.3. Preparació de l'entorn

Per poder compilar i desplegar els serveis web desenvolupats, s'ha de disposar de les següents variables d'entorn. Això permetrà a les eines d'Axis tenir accés a les llibreries necessàries.

Unix:

```
$ set AXIS_HOME=/usr/axis
$ set AXIS_LIB=$AXIS_HOME/lib
$ set AXISCLASSPATH=$AXIS_LIB/axis.jar:$AXIS_LIB/commons-discovery-
0.2.jar:$AXIS_LIB/commons-logging-
1.0.4.jar:$AXIS_LIB/jaxrpc.jar:$AXIS_LIB/saaj.jar:  $AXIS_LIB/log4j-
1.2.8.jar:$AXIS_LIB/xml-
apis.jar:$AXIS_LIB/xercesImpl.jar:$AXIS_LIB/activation.jar:$AXIS_LIB
/mail.jar

& export AXIS_HOME; export AXIS_LIB; export AXISCLASSPATH
```

Windows:

Les variables d'entorn es poden editar en el Panell de Control.

S'hauran d'afegir les variables d'entorn AXIS_HOME, AXIS_LIB i AXISCLASSPATH.

La variable AXIS_HOME ha d'apuntar al directori d'instal·lació d'Axis (*tomcat/webapps/axis*).

S'ha de recordar afegir la versió de les llibreries que correspongui en la variable d'entorn AXIS_LIB.

És important editar la variable CLASSPATH i afegir-hi la ruta de la carpeta AXIS en el Tomcat (*tomcat/webapps/axis*). D'aquesta manera es podran utilitzar les eines per desplegar serveis web en Java.

També es pot fer servir un fitxer build.xml d'Apache Ant per automatitzar les operacions necessàries per desplegar serveis web.

C.4. Creació de serveis web

El procés a seguir es basa en tres passos:

1- Primer es disposa d'un codi Java que realitza les operacions desitjades.

2- Després es desitja que aquest codi sigui accessible remotament mitjançant serveis web. Per això s'ha de desplegar en l'Apache Axis.

3- Un cop el servei web funciona en la part de servidor és necessari crear una interfície pública perquè el client pugui comunicar-se amb el servei. Aquesta interfície es tracta d'un arxiu XML amb els mètodes accessibles pels clients. La definició de la interfície es basa en *Web Service Description Language* (WSDL).

En aquesta secció s'il·lustraran els passos a seguir per crear el servei web amb un exemple. L'exemple implementat és una calculadora que ofereix les operacions de suma i resta de nombres enters.

C.4.1. Escriure el codi Java

El primer pas és implementar el codi Java que formarà el servei web. La classe d'exemple implementa dos mètodes, suma i resta. Aquests mètodes han de tenir la visibilitat *public* perquè siguin accessibles mitjançant el servei web.

Tot seguit es mostra la classe Calculadora utilitzada:

```
public class Calculadora
{
          // realitza la suma de dos nombres enters
          public int suma(int x, int y)
      {
          return x + y;
      }

      // realitza la resta de dos nombres enters
      public int resta(int x, int y)
      {
          return x – y;
      }
}
```

Un cop escrit el codi Java es compila i es copia el package amb els .class en el directori "*tomcat/webapps/axis/*"

C.4.2. Java2WSDL

En aquest punt s'ha de generar l'arxiu WSDL a partir del codi Java. Axis proporciona la eina Java2WSDL per fer això. La informació necessària per aquesta eina és la següent:

- Nom del fitxer WSDL (serveiCalculadora.wsdl)
- URL del servei web : (http://localhost:8080/axis/services/serveiCalculadora)
- Target namespace pel WSDL (urn:serveiCalculadora)
- Mapeig del package de Java que conté la classe Calculadora: (serveiCalculadora = urn:serveiCalculadora)
- Nom de la classe principal que formarà el servei web: (serveiCalculadora.Calculadora)

Per executar la eina Java2WSDL s'utilitza la següent comanda (en Windows):

```
C:\tomcat5\webapps\axis>java org.apache.axis.wsdl.Java2WSDL -o
serveiCalculadora.wsdl -
l"http://localhost:8080/axis/services/serveiCalculadora" -n
urn:serveiCalculadora -p"serveiCalculadora" urn:serveiCalculadora
serveiCalculadora.Calculadora
```

Un cop executada la comanda es genera l'arxiu WSDL amb aquest format:

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="urn:serveiCalculadora"
xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:impl="urn:serveiCalculadora" xmlns:intf="urn:serveiCalculadora"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<!--WSDL created by Apache Axis version: 1.4
Built on Apr 22, 2006 (06:55:48 PDT)-->

<wsdl:message name="sumaRequest">
      <wsdl:part name="x" type="xsd:int"/>
      <wsdl:part name="y" type="xsd:int"/>
   </wsdl:message>
   <wsdl:message name="restaRequest">
      <wsdl:part name="x" type="xsd:int"/>
      <wsdl:part name="y" type="xsd:int"/>
   </wsdl:message>
   <wsdl:message name="restaResponse">
      <wsdl:part name="restaReturn" type="xsd:int"/>
   </wsdl:message>
   <wsdl:message name="sumaResponse">
      <wsdl:part name="sumaReturn" type="xsd:int"/>
   </wsdl:message>
   <wsdl:portType name="Calculadora">
      <wsdl:operation name="suma" parameterOrder="x y">
         <wsdl:input message="impl:sumaRequest" name="sumaRequest"/>
         <wsdl:output message="impl:sumaResponse" name="sumaResponse"/>
      </wsdl:operation>
      <wsdl:operation name="resta" parameterOrder="x y">
```

```
         <wsdl:input message="impl:restaRequest" name="restaRequest"/>
         <wsdl:output message="impl:restaResponse" name="restaResponse"/>
      </wsdl:operation>
   </wsdl:portType>
   <wsdl:binding name="serveiCalculadoraSoapBinding" type="impl:Calculadora">
      <wsdlsoap:binding style="rpc"
transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="suma">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="sumaRequest">
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:serveiCalculadora" use="encoded"/>
         </wsdl:input>
         <wsdl:output name="sumaResponse">
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:serveiCalculadora" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
      <wsdl:operation name="resta">
         <wsdlsoap:operation soapAction=""/>
         <wsdl:input name="restaRequest">
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:serveiCalculadora" use="encoded"/>
         </wsdl:input>
         <wsdl:output name="restaResponse">
            <wsdlsoap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
namespace="urn:serveiCalculadora" use="encoded"/>
         </wsdl:output>
      </wsdl:operation>
   </wsdl:binding>
   <wsdl:service name="CalculadoraService">
      <wsdl:port binding="impl:serveiCalculadoraSoapBinding"
name="serveiCalculadora">
         <wsdlsoap:address
location="http://localhost:8080/axis/services/serveiCalculadora"/>
      </wsdl:port>
   </wsdl:service>
</wsdl:definitions>
```

C.4.3. WSDL2Java

Tot seguit s'explicarà com crear els stubs de comunicació, els fitxers que el client necessita per accedir al servei web.

Per això es generarà el codi en un package separat, l'anomenarem *pCalculadora*. La eina WSDL2Java necessita la següent informació:

- Directori destí base (.)
- L'abast del desplegament (Application, Request o Session)
- Habilitar el server-side generation
- Package de destí (pCalculadora)
- Nom del fitxer WSDL (serveiCalculadora.wsdl)

La comanda a executar és la següent (en Windows):

```
C:\tomcat5\webapps\axis>java org.apache.axis.wsdl.WSDL2Java --server-
side --skeletonDeploy true -p pCalculadora serveiCalculadora.wsdl
```

Després d'executar aquesta comanda es crea la carpeta *pCalculadora*. Dins d'aquesta s'hi troben els següents arxius:

- `ServeiCalculadoraSoapBindingImpl.java`: Aquesta classe és la implementació del codi del servei web. S'haurà d'editar perquè realitzi les funcions implementades en el servei web.
- `Calculadora.java`: interfície remota.
- `CalculadoraService.java`: interfície Service pels serveis web.
- `CalculadoraServiceLocator.java`: classe que utilitzarà el client per localitzar els serveis web.
- `ServeiCalculadoraSoapBindingSkeleton.java`: codi de la part de servidor.
- `ServeiCalculadoraSoapBindingStub.java`: stub de client que encapsula l'accés al servei web.
- `deploy.wsdd`: descriptor de desplegament del servei web en Axis.
- `undeploy.wsdd:` descriptor per retirar el servei web.

Ara s'ha d'editar el fitxer `ServeiCalculadoraSoapBindingImpl.java` per lligar la implementació generada del servei amb la implementació requerida del servei web. Les línies de codi afegides es poden veure ressaltades:

```
/**
 * ServeiCalculadoraSoapBindingImpl.java
 *
 * This file was auto-generated from WSDL
 * by the Apache Axis 1.4 Apr 22, 2006 (06:55:48 PDT) WSDL2Java emitter.
 */

package pCalculadora;

import serveiCalculadora.Calculadora;

public class ServeiCalculadoraSoapBindingImpl implements
pCalculadora.Calculadora{

Calculadora cal;
    public int suma(int x, int y) throws java.rmi.RemoteException {

       cal = new Calculadora();
       return cal.suma(x,y);
    }

    public int resta(int x, int y) throws java.rmi.RemoteException {

       cal = new Calculadora();
       return cal.resta(x,y);
    }

}
```

Un cop afegit el codi, s'ha de compilar tot el package *pCalculadora* i recordar copiar-lo en el directori de l'Axis.

C.4.4. Desplegar el servei web en el servidor d'aplicacions

Arribats a aquest punt ja es pot desplegar el servei web en el servidor d'aplicacions. Primer s'ha de generar un fitxer .jar a partir de les classes del servei web. Si s'utilitza un IDE com l'Eclipse, té l'opció "Export" que permet generar fitxers JAR a partir d'un projecte. També es pot fer per consola:

```
jar cvf serveiCalculadora.jar serveiCalculadora/*.class
pCalculadora/*.class
```

El fitxer resultant (serveiCalculadora.jar) s'ha de moure al directori de l'Axis en el Tomcat.

```
mv Gestora.jar tomcat/webapps/axis/WEB-INF/lib
```

Quan el fitxer JAR es trobi en el directori convenient ja es pot desplegar el servei web. S'utilitza la següent comanda:

```
C:\tomcat5\webapps\axis>java org.apache.axis.client.AdminClient
pCalculadora/deploy.wsdd
```

Si la operació ha tingut èxit es mostrarà el següent missatge per consola:

```
Processing file pCalculadora/deploy.wsdd
<Admin>Done processing</Admin>
```

Accedint a la pàgina d'inici d'Axis es pot comprovar si el servei s'ha desplegat correctament.

http://localhost:8080/axis/

Si es selecciona la opció *List*, mostrarà la llista de serveis web desplegats en el servidor d'aplicacions. Si es clica sobre (wsdl) es pot veure l'arxiu WSDL que defineix el servei web.

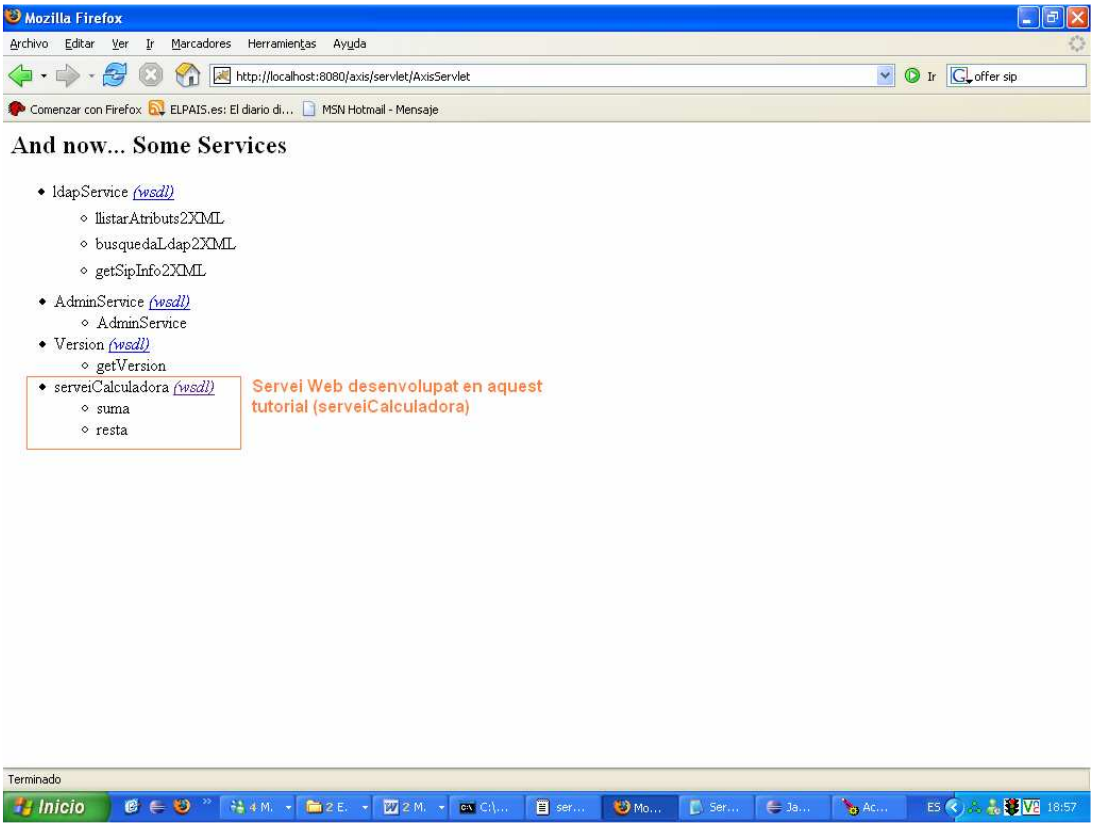


Fig. C.0.1 Llista de serveis web desplegats

Si es vol el des habilitar el servei web desplegat només caldrà executar aquesta comanda:

```
C:\tomcat5\webapps\axis>java org.apache.axis.client.AdminClient
pCalculadora/undeploy.wsdd
```

Si la operació ha tingut èxit es mostrarà el següent missatge per consola:

```
Processing file pCalculadora/undeploy.wsdd
<Admin>Done processing</Admin>
```

C.5. Implementació d'un client de servei web en Java

Per tal d'accedir al servei web des de qualsevol aplicació s'ha de desenvolupar un client que l'invoqui. En aquest cas el client també s'ha desenvolupat en Java. Els serveis web proporcionen independència de plataforma i de llenguatge de programació, per tant, el client podria desenvolupar-se en C, per exemple.

Per desenvolupar el client en Java s'ha creat un projecte amb l'Eclipse i s'ha afegit el package *pCalculadora* dins del projecte. S'ha creat un altre package amb la classe *ClientCalculadora*.

```
package clientCalculadora;
import pCalculadora.*;

public class ClientCalculadora {

  public static void main(String[] args)
  {
    CalculadoraService service = new CalculadoraServiceLocator();
    Calculadora port = null;

    System.out.println("Servei web en l'adreça: " +
    service.getserveiCalculadoraAddress());

    try{

       port = service.getserveiCalculadora();
       System.out.println("Resposta del WS a la suma 150+385: " +
                          port.suma(150, 385));

       System.out.println("Resposta del WS a la resta 77-612: " +
                          port.resta(77, 612));

       }catch(Exception e){
              e.printStackTrace();
       }
  }
}
```

Si s'executa el client mostra el següent resultat per pantalla:

```
Servei web en l'adreça:
http://localhost:8080/axis/services/serveiCalculadora

Resposta del WS a la suma 150+385: 535

Resposta del WS a la resta 77-612:-535
```

ANNEX D. DISSENY DEL PROTOCOL DE COMUNICACIÓ

La realització d'aquest treball ha suposat el disseny d'un protocol de comunicació basat en XML per comunicar el client Flash amb el servidor Java. Per al disseny d'aquests missatges s'ha tingut en compte la utilitat de cadascun i la senzillesa per disminuir el temps de processament, s'ha de recordar que l'eficiència no és precisament un avantatge de l'XML.

Per processar els missatges rebuts i construir els missatges a enviar, s'ha utilitzat un parsejador d'XML. Aquest parsejador no és més que una classe amb mètodes capaços de construir l'XML indicat segons els paràmetres d'entrada i processar XML, extraient les dades que interessen. Aquesta classe està implementada tant en Java pel WiCat Server com en ActionScript pel client.

Com s'ha explicat en el capítol 3.2.1.2 existeixen dos tipus de missatges. Els que corresponen amb peticions o respostes SIP i els que formen part dels missatges de control de l'aplicació. Els primers tenen com a tag principal *SipMessage* i els segons *DataMessage*. Com es podrà veure a continuació, tots dos disposen d'un tag intern *message_type* que especifica el tipus de missatge que contenen.

A continuació es mostren els missatges intercanviats entre client i servidor, segons les diverses situacions que es produeixen durant l'ús de l'aplicació.

D.1. Inicialització

El primer missatge que s'envia ho fa el client per autenticar a l'usuari i segueix el format següent:

```
<?xml version='1.0' encoding='iso-8859-1'?>
<DataMessage>
     <message_type>AutenticateUser</message_type>
     <username>Joan</username>
     <password>1234</password>
</DataMessage>
```

En alguns casos si el missatge enviat requereix una resposta, per simplificació es segueix el mateix format. A continuació es pot veure la resposta al missatge d'autenticació.

```
<?xml version='1.0' encoding='iso-8859-1'?>
<DataMessage>
     <message_type>AutenticateUser</message_type>
     <valid_user>true</valid_user>
</DataMessage>
```

Seguidament, si l'usuari és vàlid, el servidor envia la llista d'usuaris amb equip de videoconferència, amb tots els seus atributs:

```
<?xml version='1.0' encoding='iso-8859-1'?>
<DataMessage>
     <message_type>list_sipInfo</message_type>
     <status>OK</status>
     <ldapResult>
         <name>Ramon</name>
         <attribute id="mail”>ramon@i2cat.cat</attribute>
         <attribute id="telephoneNumber”>666555444</attribute>
         <sipUri>ramon@proxy.com</sipUri>
     </ldapResult>
     <ldapResult> ... </ldapResult>
     ...
</DataMessage>
```

Un cop l'autenticació ha tingut èxit s'envia un missatge de tipus SIP *register* per registrar l'usuari en el proxy SIP registrar:

```
<?xml version='1.0' encoding='iso-8859-1'?>
<SipMessage>
     <message_type>register</message_type>
     <username>Joan</username>
     <sipUri>joan@proxy.com</sipUri>
</SipMessage>
```

Les respostes SIP *response* també es reenvien cap al client Flash, els missatges d'aquest tipus segueixen el mateix format:

```
<?xml version='1.0' encoding='iso-8859-1'?>
<SipMessage>
     <message_type>response</message_type>
     <sender_client>Joan</sender_client>
     <receiver_client>proxy</receiver_client>
     <reason>register</reason>
     <status>200 OK</status>
</SipMessage>
```

Després del registre toca la subscripció al Presence Agent, el missatge que s'envia és com el següent:

```
<?xml version='1.0' encoding='iso-8859-1'?>
<SipMessage>
     <message_type>subscribe</message_type>
     <sender_client>Joan</sender_client>
     <receiver_client>PresenceAgent</receiver_client>
     <event>presence</event>
</SipMessage>
```

El Presence Agent subscriu a l'usuari i envia un NOTIFY cap al B2BUA amb la llista d'usuaris subscrits a l'event *presence.* El corresponent NOTIFY amb el protocol propi, que s'envia al client és el següent:

```
<?xml version='1.0' encoding='iso-8859-1'?>
<SipMessage>
     <message_type>notify</message_type>
     <sender_client>PresenceAgent</sender_client>
     <receiver_client>Joan</receiver_client>
     <event>presence</event>
     <user>
       <name>Ramon</name>
       <status>online</status>
     </user>
     <user>
       <name>Alicia</name>
       <status>offline</status>
     </user>
     ...
</SipMessage>
```

D.2. **Programació d'una videoconferència**

Quan un usuari clica el botó de "programar videoconferències" en la interfície web, haurà d'emplenar un formulari. Aquest formulari consta d'un calendari per seleccionar la data, un selector per la hora i pel minut, una llista on es trien els usuaris implicats i un quadre de text on s'introdueix la descripció de la videoconferència.

Després d'emplenar el formulari i prémer el botó "programar" s'envia cap al gestor de videoconferències un missatge com el següent:

```
<?xml version='1.0' encoding='iso-8859-1'?>
<DataMessage>
     <message_type>newSession</message_type>
     <description>Reunió departament de ventes</description>
     <user_master>Joan</user_master>
     <user>Alicia</user>
     <user>Ramon</user>
     <year>2007</year>
     <month>02</month>
     <day>20</day>
     <hour>16</hour>
     <minute>30</minute>
</DataMessage>
```

El gestor de videoconferències crea una nova sessió amb les característiques d'aquesta, li assigna un identificador i l'emmagatzema fins que és l'hora d'iniciar-la.

D.3. Consultar videoconferències programades

Quan un usuari clica el botó “consultar” en el panell “consultar videoconferències”, s’envia un missatge de consulta com el següent dirigit al gestor de videoconferències:

```
<?xml version='1.0' encoding='iso-8859-1'?>
<DataMessage>
     <message_type>requestSessions</message_type>
     <user>Joan</user>
</DataMessage>
```

El gestor de videoconferències consultarà totes les videoconferències programades on aparegui l’usuari Joan. Si una d’elles està en curs en aquell moment enviarà un XML com el següent a l’usuari Joan:

```
<?xml version='1.0' encoding='iso-8859-1'?>
<DataMessage>
     <message_type>actualSession</message_type>
     <description>Reunió d’investigadors</description>
     <id>60</id>
     <user_master>Lluis</user_master>
     <user>Joan</user>
     <user>Alicia</user>
     <user>Ramon</user>
</DataMessage>
```

Si no hi ha cap videoconferència en curs on l’usuari Joan estigui convidat, o si l’usuari no accepta participar-hi s’enviarà un XML amb aquest format:

```
<?xml version='1.0' encoding='iso-8859-1'?>
<DataMessage>
     <message_type>userSessions</message_type>
     <session>
         <description>Reunió</description>
         <user_master>Alicia</user_master>
         <user>Joan</user>
         <user>Ramon</user>
         <year>2007</year>
         <month>02</month>
         <day>20</day>
         <hour>16</hour>
         <minute>30</minute>
     </session>
     <session> ... </session>
     ...
</DataMessage>
```

Si no existeix cap videoconferència programada en la qual l'usuari Joan estigui convidat, s'enviarà el mateix XML però sense cap etiqueta *session*.

D.4. Trucada simple

En el cas d'una trucada de dos usuaris els missatges intercanviats entre client i servidor són els següents:

Suposem que l'usuari Joan inicia una trucada amb l'usuària Alicia:

```
<?xml version='1.0' encoding='iso-8859-1'?>
<SipMessage>
     <message_type>invite</message_type>
     <sender_client>Joan</sender_client>
     <receiver_client>Alicia</receiver_client>
</SipMessage>
```

La resposta del B2BUA de l'Alicia es redirecciona cap al client Flash amb el següent format:

```
<?xml version='1.0' encoding='iso-8859-1'?>
<SipMessage>
     <message_type>response</message_type>
     <sender_client>Joan</sender_client>
     <receiver_client>Alicia</receiver_client>
     <reason>invite</reason>
     <status>200 OK</status>
</SipMessage>
```

La trucada ja està iniciada, suposem que al cap d'una estona l'usuària Alicia vol acabar la trucada:

```
<?xml version='1.0' encoding='iso-8859-1'?>
<SipMessage>
     <message_type>bye</message_type>
     <sender_client>Alicia</sender_client>
     <receiver_client>Joan</receiver_client>
</SipMessage>
```

Al rebre la resposta el client Flash de l'Alicia acabarà la videoconferència:

```
<?xml version='1.0' encoding='iso-8859-1'?>
<SipMessage>
     <message_type>response</message_type>
     <sender_client>Alicia</sender_client>
     <receiver_client>Joan</receiver_client>
     <reason>bye</reason>
     <status>200 OK</status>
</SipMessage>
```

D.5. Multi conferència senzilla

La seqüència de missatges que es mostra a continuació correspon amb la situació en la qual un usuari selecciona varis usuaris de la llista de presència i prem el botó de trucar.

Suposarem que l'usuari Joan truca als usuaris Alicia i Ramon.

```
<?xml version='1.0' encoding='iso-8859-1'?>
<DataMessage>
     <message_type>initSession</message_type>
     <id>5643</id>
     <user_master>Joan</user_master>
     <user>Alicia</user>
     <user>Ramon</user>
</DataMessage>
```

Els usuaris Alicia i Ramon reben el següent missatge del servidor:

```
<?xml version='1.0' encoding='iso-8859-1'?>
<DataMessage>
     <message_type>multiSession</message_type>
     <id>5643</id>
     <user_master>Joan</user_master>
     <user>Alicia</user>
     <user>Ramon</user>
</DataMessage>
```

El client Flash de l'usuari Joan enviarà un XML com el següent per subscriure's a la trucada, si l'Alicia i el Ramon accepten la invitació, els seus respectius clients Flash també l'enviaran:

```
<?xml version='1.0' encoding='iso-8859-1'?>
<SipMessage>
     <message_type>call_subscribe</message_type>
     <sender_client>Joan</sender_client>
     <receiver_client>PresenceAgent</receiver_client>
     <event>presence</event>
     <event_id>5643<event_id>
</SipMessage>
```

Si qualsevol d'ells vol acabar la multi conferència enviarà un missatge cap al seu corresponent B2BUA com el següent:

```
<?xml version='1.0' encoding='iso-8859-1'?>
<SipMessage>
     <message_type>call_unsubscribe</message_type>
     <sender_client>Joan</sender_client>
     <receiver_client>PresenceAgent</receiver_client>
     <event>presence</event>
     <event_id>5643<event_id>
</SipMessage>
```

D.6. **Multi conferència programada**

Quan arriba l'hora d'iniciar una multi conferència programada s'envien els missatges mostrats a continuació.

El primer missatge l'envia el gestor de videoconferències per informar a l'usuari Màster (el que ha programat la videoconferència) que és l'hora d'iniciar-la.

```
<?xml version='1.0' encoding='iso-8859-1'?>
<DataMessage>
     <message_type>initProgSession</message_type>
     <id>40</id>
     <description>Reunió departament de ventes</description>
     <user_master>Joan</user_master>
     <user>Alicia</user>
     <user>Ramon</user>
</DataMessage>
```

Si el Màster accepta començar la videoconferència, respondrà amb un missatge igual que l'anterior per confirmar l'inici de la videoconferència.

El gestor de videoconferències s'encarregarà d'avisar a la resta d'usuaris implicats amb un missatge com el que es mostra a continuació:

```
<?xml version='1.0' encoding='iso-8859-1'?>
<DataMessage>
     <message_type>progMultiSession</message_type>
     <id>40</id>
     <description>Reunió departament de ventes</description>
     <user_master>Joan</user_master>
     <user>Alicia</user>
     <user>Ramon</user>
</DataMessage>
```

El client Flash de l'usuari que rebi aquest missatge mostrarà una alerta amb la informació de la multi conferència. Si l'usuari accepta participar-hi enviarà un missatge "call_subscribe" com en el cas de les multi conferències senzilles. El Màster també enviarà aquest missatge.

Quan l'usuari Màster vulgui sortir de la multi conferència, clicarà el botó de penjar i el seu client Flash enviarà el següent XML:

```
<?xml version='1.0' encoding='iso-8859-1'?>
<DataMessage>
     <message_type>deleteSession</message_type>
     <id>40</id>
</DataMessage>
```

Així el gestor de videoconferències eliminarà la informació d'aquesta multi conferència, considerada com finalitzada.